UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de grado

Diseño y desarrollo de un control inteligente para una sala de estimulación multisensorial de bajo coste

Lucía Fernández Torres

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Diseño y desarrollo de un control inteligente para una sala de estimulación multisensorial de bajo coste

Autor: Lucía Fernández Torres
Tutor: Javier Gómez Escribano

mayo 2021

Lucía Fernández Torres
Diseño y desarrollo de un control inteligente para una sala de estimulación multisensorial de bajo coste

Lucía Fernández Torres
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos

Agradecimientos

En primer lugar, gracias a mi tutor Javier Gómez Escribano, por acompañarme y guiarme durante todos estos meses de trabajo, a la par que dejándome dar rienda suelta a mi imaginación.

A mi familia, mi pilar incondicional. Mamá, Papá, gracias por haber luchado tanto para poder darme una educación de calidad y ayudarme a cumplir todos mis objetivos. A David, por su ayuda y paciencia, y a mi mayor referente, mi hermana Gema. Una mujer fuerte, inteligente y luchadora que siempre me recuerda que mis límites están donde yo quiero que estén.

Por último, pero no menos importante, gracias a mis amigos de la Universidad, con quienes he compartido tantas horas en los laboratorios y risas en la cafetería. Ayala, Mihai, ya sois parte de mi familia.

Resumen

Este Trabajo de Fin de Grado se centra en el desarrollo de un software que facilite el control de dispositivos inteligentes, exponiendo la posibilidad de generar un espacio en el que se integren diferentes tecnologías y dispositivos electrónicos para su uso terapéutico.

El principal objetivo de este proyecto es el desarrollo de un software de control que permita poder interactuar con diferentes dispositivos en tiempo real, así como poder programar secuencias de acciones y generar sesiones que puedan ser almacenadas y reproducidas posteriormente. De esta forma, se pretende crear un entorno para la estimulación sensorial, visual y auditiva, de niños y adultos, y además poder controlar la reacción a estos estímulos con la manipulación y modificación de dichas sesiones en tiempo real.

Para ello, se ha creado una aplicación compatible para dispositivos Android, aprovechando e integrando las capacidades de diversas tecnologías hardware y software. En este trabajo se proporcionan los requisitos y pautas a seguir para configurar este entorno y que puedan utilizarlo tanto los terapeutas encargados de organizar las sesiones, como los usuarios finales que deseen utilizar el entorno previamente configurado.

Palabras clave

Estimulación multisensorial, educación especial, dispositivos IoT, automatización, Android, entornos inteligentes

Abstract

This Degree Final Project is focused on the development of a software which facilitates the control of smart devices, exposing the possibility of generating a space in which different devices and technologies are integrated for its therapeutic use.

The main object of this project is the development of a control software that allows the user to interact with different devices in real time, as well as to be able to program sequences of actions and generate sessions that can be stored and reproduced later. In this way, it is intended to create an environment for sensory, visual and auditory stimulation of children and adults, and also to be able to control the reaction to these stimuli with the manipulation and modification of said sessions in real time.

For this purpose, a compatible application for Android devices has been created, taking advantage and integrating the capabilities of various hardware and software technologies. In this project, the requirements and guidelines to follow in order to configure this environment are provided, in such a way that it can be used by the therapists in charge of organising the sessions, as well as by the end users who wish to use the previously configured environment.

Keywords

Multisensory stimulation, special education, IoT devices, automation, Android, smart environments

Índice

1 Introducción **1**

1.1 Motivación . . . 1

1.2 Objetivos . . . 1

1.3 Organización de la memoria . . . 2

2 Estado del Arte **3**

2.1 Salas Multisensoriales . . . 3

2.1.1 *DOIT Sensory Console* . . . 4

2.1.2 Tecnología *Eneso Sense* . . . 5

2.1.3 BJ Adaptaciones . . . 5

2.2 Tecnologías para el desarrollo de entornos inteligentes . . . 6

2.2.1 OpenHAB . . . 7

2.2.2 *Home Assistant* . . . 7

3 Análisis y diseño **9**

3.1 Dispositivos y tecnologías utilizadas . . . 9

3.2 Requisitos del sistema . . . 10

3.2.1 *Requisitos Funcionales* . . . 10

3.2.2 *Requisitos No Funcionales* . . . 13

3.3 Diseño de la aplicación . . . 14

3.3.1 *Arquitectura del Sistema* . . . 14

3.3.2 *Lógica del Sistema* . . . 14

3.3.3 *Diseño de la interfaz de usuario* . . . 15

4 Despliegue **21**

4.1 Despliegue de *Home Assistant* . . . 21

4.2 Integración de dispositivos . . . 22

5 Implementación **23**

5.1 Configuración del sistema . . . 23

5.2 Conexión con *Home Assistant* . . . 25

5.3 Interacción con los dispositivos inteligentes . . . 25

5.4 Almacenamiento de sesiones . . . 28

5.5 Reproducción de secuencias de acciones . . . 30

6 Conclusiones y trabajo futuro **33**

6.1 Conclusiones 33
6.2 Trabajo futuro 34

Bibliografía **36**

Apéndices **37**

A Diagrama de clases de la aplicación **39**

B Instalación y despliegue de *Home Assistant* **41**

B.1 Requisitos para la instalación 41
B.2 Descarga de la imagen 41
B.3 Puesta en marcha de la Raspberry Pi 43
B.4 Configuración de *Home Assistant* 43

C Integración de dispositivos **45**

C.1 Luz Philips Hue 45
C.2 Google Home 47
C.3 Enchufe inteligente Xiaomi 47

Listas

Lista de códigos

5.1 Fichero de configuración JSON 24
5.2 Método *onCreate del fragmento inicial* 24
5.3 Instancia de *HomeAssistantConfig* 25
5.4 Instancia de *HAConnector a partir de la configuración leída* 25
5.5 Proceso de creación de los dispositivos 26
5.6 Comprobación del estado del servidor de *Home Assistant* 27
5.7 Creación de un *ActionableFunction* para la función *setEffect* de la clase *Light* 28
5.8 Ejemplo llamada *rgbColor* 29
5.9 Función para añadir acciones a una sesión 29
5.10 Almacenamiento y recuperación de sesiones 30
5.11 Pseudocódigo función que inicia la reproducción de una sesión: *run()* 31
5.12 Pseudocódigo función que pausa la reproducción de una sesión: *pause()* 32
5.13 Pseudocódigo función que finaliza la reproducción de una sesión: *stop()* 32

Lista de figuras

2.1 *DOIT Sensory Console*: Panel principal 4
2.2 Sala Lite Sense 5
2.3 Software de control del Sistema SHX 6
2.4 Logo y consola principal de *OpenHAB* 7
2.5 Logo *Home Assistant* 7

3.1 Arquitectura del Sistema 14
3.2 Pantalla principal 16
3.3 Pantalla *Play Sesión* 16
3.4 Pantalla listar sesiones 16
3.5 Pantallas grabar sesión 17
3.6 Pantallas de visualización de una sesión 18
3.7 Pantallas guardar sesión 18
3.8 Pantalla añadir acción 19
3.9 Pantalla sesión con acciones añadidas 19

3.10 Pantalla eliminar acción 20

4.1 Integración de dispositivos en *Home Assistant* 22

5.1 Diagrama de secuencia: Cambiar de color luz RGB 29

A.1 Diagrama de las clases más representativas que conforman *MyPlayRoom* 40

B.1 Descarga de la imagen 42
B.2 Selección de la tarjeta SD 42
B.3 Finalización del proceso 42
B.4 Creación de una cuenta de usuario 43
B.5 Nombre de la cuenta 44
B.6 Finalización de la configuración de *Home Assistant* 44

C.1 Integración Luz Philips: Descubrimiento del dispositivo 45
C.2 Integración Luz Philips: Instalación en curso 46
C.3 Integración Luz Philips: Configuración del *Hue* 46
C.4 Integración Luz Philips: Fin del proceso 46
C.5 Integración Google Home: Descubrimiento del dispositivo 47
C.6 Integración Google Home: Confirmación del usuario y asignación de un área 47
C.7 Integración Google Home: Fin del proceso 48
C.8 Integración Enchufe Xiaomi: Descarga de la aplicación *Mi Home* 48
C.9 Integración Enchufe Xiaomi: Cambio de región a China 48
C.10 Integración Enchufe Xiaomi: Escaneo de dispositivos 49
C.11 Integración Enchufe Xiaomi: Conexión 49
C.12 Integración Enchufe Xiaomi: Inclusión del dispositivo 50
C.13 Integración Enchufe Xiaomi: Extracción del token 50
C.14 Integración Enchufe Xiaomi: Introducción de los valores requeridos 51
C.15 Integración Enchufe Xiaomi: Fin del proceso 51

1

Introducción

1.1. Motivación

La motivación de este trabajo nace del deseo de desarrollar un sistema inteligente para el control de una sala de estimulación sensorial diseñada y desarrollada con elementos de bajo coste, que permita a los terapeutas controlar el ambiente de la misma y regular los estímulos a los que se someten los usuarios finales, accesible para todo tipo de profesionales y usuarios que deseen experimentar con la herramienta.

1.2. Objetivos

El objetivo principal de este trabajo es el diseño y desarrollo de un prototipo hardware y software que permita por un lado, la estimulación visual y auditiva de los usuarios finales otorgándoles la posibilidad de interactuar en tiempo real con dispositivos inteligentes y por otro lado, otorgar a los profesionales la posibilidad de configurar sesiones personalizadas para cada usuario en base a sus necesidades y poder almacenarlas para su utilización futura:

- Diseñando y configurando un entorno de pruebas habilitado para el uso de diversos dispositivos inteligentes en un marco terapéutico.
- Desarrollando un sistema de control que permita interactuar con dispositivos inteligentes en tiempo real, así como programar secuencias de acciones personalizadas y la capacidad de almacenar sesiones para su uso posterior.
- Creando un prototipo fácilmente escalable, que permita la integración y el control futuro de nuevos dispositivos.
- Diseñando de una interfaz gráfica sencilla y visualmente atractiva.

1.3. Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estudio del estado del arte**. En este capítulo se introducirán las tecnologías y dispositivos utilizados para la elaboración de este proyecto y se contextualizará el uso actual de las salas multisensoriales como herramienta terapéutica.
- **Análisis y diseño**. En esta sección se discutirán las diferentes decisiones de diseño tomadas en la definición y desarrollo del proyecto.
- **Despliegue**. En este capítulo se explicará cómo se ha realizado el despliegue del servidor de *Home Assistant* y cómo se ha realizado la integración de los dispositivos utilizados en el prototipo de este proyecto.
- **Implementación**. En esta sección se explicarán detalladamente las diferentes fases de creación del prototipo, incluyendo la configuración de los diferentes dispositivos y la inclusión de funcionalidades en el sistema de control.
- **Conclusiones y trabajo futuro**. Por último, se analizará el prototipo creado y se discutirán posibles mejoras e ideas para la integración de nueva funcionalidad en trabajos futuros.

2

Estado del Arte

En este capítulo se introducen los conceptos y herramientas utilizados para la realización de este Trabajo de Fin de Grado (TFG). En primer lugar, se explican los beneficios de la utilización de salas multisensoriales como herramienta terapéutica y se muestra un análisis de las soluciones actualmente existentes.

A continuación, se analizan diferentes tecnologías que permiten el desarrollo de entornos inteligentes, espacios en los que se utiliza la tecnología de los sistemas embebidos, así como otras tecnologías de la información y la comunicación empleadas para la creación de ambientes interactivos que acercan la computación al mundo físico y a los problemas cotidianos.

Finalmente, se muestran las conclusiones obtenidas tras el análisis del Estado del Arte y las decisiones tomadas para proceder con el diseño y desarrollo del TFG.

2.1. Salas Multisensoriales

Las Salas Multisensoriales son *"entornos en los que se busca trabajar a nivel físico y cognitivo con sus usuarios, utilizando la estimulación controlada y regulable de los diferentes sentidos como medio para alcanzar múltiples objetivos de aprendizaje, terapéuticos, lúdicos, comunicativos, de relajación y de bienestar"* [1]. Según afirman los expertos, en estas salas *"es posible trabajar desde muy diversos enfoques y abordar objetivos muy dispares teniendo en cuenta a los distintos usuarios y sus necesidades concretas"*.

Además, permiten reducir de manera progresiva y de forma no invasora la necesidad de sobreestimulación sensorial que pueden presentar algunas personas, permitiendo aislar los diferentes estímulos e ir sumando progresivamente y de forma coherente aquellos estímulos que el usuario tolera y considera reconfortantes, así como permitiendo identificar sensaciones adversas.

Otro aspecto interesante de las salas multisensoriales es su papel como reforzador positivo, ya que se trata de una herramienta que puede ser utilizada para trabajar la comunicación y el lenguaje expresivo. El usuario puede encontrar muy gratificante la activación de un estímulo o elemento de la sala

y se puede aprovechar esta situación *"ofreciendo un entorno coherente para convertir las peticiones del usuario en acciones motivadoras"* [2] . Es decir, el usuario que está disfrutando de la sala puede realizar peticiones al profesional encargado del control de la sala, de tal forma que se puede trabajar además la comunicación.

Además de la versatilidad de este tipo de salas para adaptarse a cualquier persona y objetivo, ya sea de tipo terapéutico, conductual, académico, etc., se pueden utilizar en entornos lúdicos. El ocio y el bienestar son objetivos en sí mismos y estas salas permiten ocupar el tiempo libre con actividades que fomenten sensaciones agradables y la autonomía.

En este apartado se van a exponer las soluciones que diferentes proveedores ofrecen para instalar y capacitar salas multisensoriales en España, realizándose un análisis de las tecnologias que ofrecen y una estimación del presupuesto de las mismas.

2.1.1. *DOIT Sensory Console*

La compañía DOIT [3] ofrece un amplio catálogo de materiales sensoriales y salas multisensoriales diseñadas a medida para cubrir todo tipo de necesidades. La compañía utiliza un software propio, *DOIT Sensory Console* [4], para el control de todos los componentes del elemento multisensorial, permitiendo al usuario interactuar con ellos y realizar múltiples actividades cognitivas.

Figura 2.1: DOIT Sensory Console: Panel principal

El entorno que esta compañía propone utiliza de base una *Tablet* para manipular el software de control, un proyector o pantalla, un equipo de audio y diversos dispositivos multisensoriales conectados utilizando el *DOIT Adapter*. El software de control únicamente permite manipular dispositivos multisensoriales de la gama *DOIT*, ya que para la integración de dispositivos se utiliza un protocolo de comunicación propio (*DOITLink*) [5]. Respecto a los dispositivos de otras marcas, únicamente permiten su encendido y apagado.

A modo de estimación, la implementación de una sala compuesta por los dispositivos mencionados a continuación controlados a través de la *DOIT Sensory Console* puede tener un coste aproximado de 30.000€ [6], con la instalación incluida de: dos columnas interactivas –una con burbujas y otra con una

circulación de pelotas en su interior–, una caja compuesta por 8 pulsadores de colores, una alfombra con puntos de luz, una escalera de colores y un panel de luz que reaccionan a la voz, una piscina de bolas con luz y una superficie *viva* que reacciona al movimiento.

2.1.2. Tecnología *Eneso Sense*

Otra compañía que ofrece un servicio similar es *Eneso* [7], con la utilización de su software *Eneso Sense* [8] para la integración e interacción con dispositivos sensoriales. La alternativa más económica que ofrecen es la sala *Lite Sense*, cuyo precio aproximado es de 4.000€.

Figura 2.2: Sala *Lite Sense* de *Eneso*

Esta sala incluye los siguientes elementos: un tubo de burbujas interactivo con una abrazadera y un pedestal incluidos, dos espejos acrílicos de seguridad para multiplicar el efecto del tubo de burbujas, un mazo de fibra óptica, una botonera y una bomba de vaciado para el tubo de burbujas [9].

En cuanto a las capacidades de este software de control, además de la posibilidad de interactuar con los elementos, destaca la capacidad de combinar elementos de causa (como por ejemplo la botonera) y elementos de efecto (como el tubo de burbujas).

2.1.3. BJ Adaptaciones

BJ Adaptaciones [10] es otra de las compañías líderes en el diseño y desarrollo de salas de estimulación multisensorial, habiendo desplegado más de 400 salas en todo el mundo. Proporciona un servicio personalizado, diseñando las salas según las necesidades específicas de los usuarios, cuyo precio puede variar entre 3.000€ y 30.000€ según su complejidad.

La compañía destaca por ser la creadora del Sistema SHX [11], una tecnología diseñada con la intención de ir un paso más allá, haciendo que toda la sala se convierta en un nuevo mundo temático

con solo pulsar un botón, haciendo que diferentes elementos interactúen entre sí de forma coordinada. Al igual que en el la solución desarrollada por *DOIT*, todos los dispositivos que se pueden integrar en la sala se controlan a través de un software que puede ser utilizado en una tablet u ordenador, como se puede observar en la Figura 2.3.

Figura 2.3: Software de control del Sistema SHX

Este tipo de salas están pensadas para que no solo el terapeuta o profesional sea quien lidere la sesión, sino que también sean los propios usuarios los que tomen el control total de la sala, de acuerdo con los objetivos terapéuticos establecidos.

2.2. Tecnologías para el desarrollo de entornos inteligentes

Con el objetivo de lograr integrar dispositivos de diversa naturaleza en el entorno desarrollado en este proyecto, se han estudiado las capacidades de plataformas *open source* comúnmente utilizadas para evaluar la idoneidad de su utilización en este proyecto.

Aunque actualmente existe en el mercado una gran variedad de dispositivos IoT, con precios cada vez más asequibles, existe un problema y una limitación a la hora de diseñar entornos domóticos, debido a la ausencia de interfaces enfocadas a usuarios con diferentes necesidades y a la baja interoperabilidad entre los fabricantes de los distintos sistemas de automatización.

Debido a estas necesidades, nacieron las plataforma domóticas de código abierto que se pueden descargar y desplegar en ordenadores de bajo coste, como puede ser una Raspberry Pi [12]. Siguiendo la filosofía *open source*, este planteamiento fomenta la colaboración y la difusión del trabajo, haciendo que las personas puedan compartir sus ideas y beneficiarse de esta retroalimentación continua.

2.2.1. OpenHAB

OpenHAB [13] (Open Home Automation Bus) es una de las herramientas de domótica más populares dentro de la comunidad *open source*. Escrito en Java, está diseñado para ser independiente del dispositivo, a la par que facilita a los desarrolladores incluir sus propios dispositivos y *plugins* al sistema. Soporta más de 200 tecnologías y sistemas y puede ejecutarse en diversos entornos: Linux, macOS, Windows, Raspberry Pi, Docker y Synology, accediendo a él con aplicaciones web, iOS y Android [14].

Se trata de una herramienta orientada a ser ejecutada sobre el hardware del usuario, de tal forma que mantiene los datos de forma privada en la red local que se ejecute. Su código fuente está disponible en GitHub, bajo la Licencia Pública Eclipse 2.0 (EPL) [15].

Figura 2.4: Logo y consola principal de *OpenHAB*

2.2.2. *Home Assistant*

Home Assistant [16] es una plataforma de código abierto desarrollada en Python para la gestión de dispositivos de domótica. Su uso está orientado al hogar, y es capaz de integrar gran cantidad de dispositivos y servicios, mostrando un alto índice de compatibilidad.

Esta plataforma ofrece una interfaz web, pero la parte más atractiva y diferencial por la cual se ha decidido su utilización en este proyecto, es que proporciona una API RESTful [17]. Esto permite su empleo a modo de *middleware* para realizar llamadas desde el software desarrollado en este proyecto y desencadenar acciones en los dispositivos integrados en ella.

Figura 2.5: Logo *Home Assistant*

3

Análisis y diseño

En esta sección se detallan los dispositivos y tecnologías utilizadas en este proyecto, los requisitos desarrollados en la fase de análisis del proyecto y se refleja el diseño de la aplicación, incluyendo la lógica del sistema y las decisiones de diseño de la interfaz de usuario.

3.1. Dispositivos y tecnologías utilizadas

Para la elaboración de este proyecto se ha decidido aprovechar las capacidades del sistema de domótica *Home Assistant* para la integración de dispositivos de diversa naturaleza, de tal forma que el software desarrollado en este proyecto permita la interacción con dichos dispositivos.

A pesar de que este prototipo está planteado de tal forma que en un futuro puedan añadirse más dispositivos y sea escalable tanto vertical como horizontalmente, para la realización de este proyecto se han seleccionado los dispositivos detallados a continuación, teniendo en cuenta sus capacidades.

Luz Philips Hue

Se trata de una lámpara de mesa RGBW de intensidad regulable, con hasta 3 horas de autonomía. Esta lámpara se conecta a un *hub*, un aparato que centraliza la comunicación entre las luces Philips Hue [18] (soportando hasta 50 lámparas y bombillas conectadas simultáneamente), y los dispositivos que lo controlan (aplicaciones *ad-hoc* o dispositivos de terceros, como Amazon Echo Dot y Google Home). Para ello, además de a la corriente, este *hub* está conectado a la red utilizando un cable Ethernet, permitiendo así la conexión Wi-Fi de las bombillas.

Existen actualmente en el mercado muchas bombillas que funcionan sin la existencia de este *hub*, conectándose simplemente a la red Wi-Fi, por lo que pueden parecer más funcionales. Sin embargo, los sistemas desplegados a través de un *hub* son mucho más eficientes y se integran mejor en ecosistemas complejos.

Altavoz inteligente *Google Home*

Se trata de un asistente inteligente con control por voz [19] que posee la capacidad de poner alarmas, recordatorios, organizar la agenda, responder a preguntas básicas como por ejemplo ofrecer las noticias o el tiempo, e incluso controlar otros dispositivos inteligentes conectados a la red Wi-Fi.

A pesar de las amplias capacidades que ofrece, en este proyecto se va a utilizar a modo de altavoz inteligente para la reproducción de sonidos y canciones demandados a través del software desarrollado.

Enchufe inteligente *Xiaomi Mi Smart Power Plug*

Se trata de un dispositivo muy versátil, ya que permite controlar cualquier aparato conectado a él, otorgando la capacidad de encendido y apagado vía Wi-Fi [20].

3.2. Requisitos del sistema

En esta sección se describen los requisitos que debe cumplir la aplicación para cubrir todas las necesidades contempladas durante la fase de diseño. Estos requisitos se han dividido en funcionales y no funcionales.

3.2.1. *Requisitos Funcionales*

Los requisitos funcionales son aquellos que hacen referencia al comportamiento de las funciones y servicios del sistema. Son necesarios para comenzar con la fase de diseño, ya que con ellos se define la estructura y el alcance del proyecto que se va a desarrollar.

A continuación se muestra el listado con la información detallada de los requisitos funcionales que debe cumplir el sistema.

RF01. Conexión con *Home Assistant*.

- Al iniciar la aplicación, se deberá pulsar un botón para comprobar la conexión con el servidor de *Home Assistant*.
- Si la conexión se ha realizado correctamente, se le mostrará un mensaje al usuario y podrá acceder al resto de funcionalidad de la aplicación.
- Si la conexión no se ha podido establecer, se abrirá una ventana emergente informando al usuario del error.

RF02. Selección del modo de uso.

- Desde la pantalla principal de la aplicación, el usuario podrá seleccionar uno de los dos modos de uso: *Play sesión* o *Grabar sesión*.
- Previamente, el usuario deberá haber pulsado el botón correspondiente para comprobar que el servidor de *Home Assistant* está activo.

RF03. Listado de dispositivos disponibles.

- Se mostrará al usuario un listado de todos los dispositivos activos y las funciones disponibles para cada uno de ellos.

RF04. Dispositivos soportados y funciones disponibles.

- La aplicación soportará dispositivos de tipo: *Light* (luz RGBW), *Socket* (enchufe inteligente) y *MediaPlayer* (reproductor de audio). Todos ellos podrán encenderse y apagarse.
- Se podrá modificar el color y la intensidad de los dispositivos de tipo *Light*, así como poder elegir dos tipos de efecto: *random* (selección aleatoria de un color) y *colorloop* (bucle de colores aleatorio).
- Se podrán activar y desactivar los enchufes inteligentes, dispositivos de tipo *Socket*.
- La aplicación contará con una biblioteca de archivos multimedia.
- Los dispositivos de tipo *MediaPlayer* dispondrán de la funcionalidad de iniciar y finalizar la reproducción de sonidos disponibles en la biblioteca de archivos.

RF05. Interacción con los dispositivos en tiempo real.

- El usuario podrá utilizar la interfaz proporcionada para interactuar con los dispositivos disponibles en tiempo real.

RF06. Crear sesión.

- El usuario podrá crear sesiones en las que se añadirán secuencias de acciones.
- Al crear la sesión, se pedirá al usuario introducir un nombre de sesión, que será único. En el caso de que ya exista una sesión con el nombre introducido, se informará al usuario y no se permitirá su creación.

RF07. Eliminar sesión.

- El usuario podrá eliminar las sesiones previamente creadas y almacenadas en el sistema.

RF08. Guardar sesión.

- Si el usuario desea guardar la sesión creada para poder modificarla o reproducirla en un futuro, deberá pulsar el botón habilitado para ello.

- A la hora de guardar una sesión, se ofrecerá al usuario la posibilidad de modificar el nomrbre de la sesión almacenada. No se podrá utilizar el nombre de una sesión ya almacenada, este identificativo deberá ser único.

RF09. Listar sesiones.

- Se mostrará al usuario todas las sesiones disponibles que han sido previamente creadas.
- Al seleccionar una sesión en concreto, se podrá acceder a su contenido.

RF10. Añadir acciones a una sesión.

- Se considerará *acción* cada una de las opciones que se pueden realizar sobre un dispositivo.
- Las secuencias de acciones se organizarán en lo que se denominan *canales*.
- Cada sesión estará compuesta por cuatro canales en total, en los que se podrán añadir secuencias de acciones de los dispositivos de manera simultánea. Uno de ellos estará dedicado a dispositivos de tipo *MediaPlayer*, proporcionando un canal de audio exclusivo. El resto, estarán dedicados a la gestión de otros dispositivos genéricos (cualquiera salvo *MediaPlayer*).
- Para añadir una acción en los canales dedicados a dispositivos genéricos, el usuario deberá seleccionar: dispositivo, segundos de inicio y fin, y acción. Para cada acción seleccionada, se mostrará la información adicional que sea necesaria introducir (por ejemplo, en el caso de seleccionar la acción *color* de una luz, se mostrará una barra para la selección de color).
- Las acciones tendrán una duración mínima de 3 segundos.
- El final de la acción deberá ser superior al inicio de la misma.
- No se podrán añadir acciones si en ese mismo canal hay una acción añadida dentro del rango de tiempo seleccionado.
- No se podrán añadir acciones para un dispositivo que está siendo utilizado en otro canal dentro del rango de tiempo seleccionado.

RF11. Eliminar acción.

- Al pulsar sobre una acción previamente añadida, se abrirá una ventana emergente en la que se dará al usuario la posibilidad de eliminarla.
- Una acción eliminada no podrá ser recuperada en un futuro.

RF12. Ejecutar sesión.

- Utilizando el botón habilitado para ello, se podrá iniciar la reproducción de una sesión almacenada o recientemente creada (sin necesidad de ser guardada).

RF13. Pausar sesión en curso.

- Se podrá pausar la sesión cuya reproducción está en curso, pudiendo ser reanudada o finalizada a partir de ese estado.

RF14. Parar sesión en curso.

- Se podrá finalizar una sesión cuya reproducción está en curso.

3.2.2. *Requisitos No Funcionales*

Los requisitos no funcionales son aquellos requisitos que imponen restricciones en el diseño o la implementación del sistema. Estas pueden ser restricciones en el tiempo, sobre estándares en el desarrollo o implementación, rendimiento del sistema o seguridad, entre otras.

A continuación se muestra el detalle de los requisitos no funcionales que debe cumplir el sistema.

RNF01. Versión de Android. El dispositivo sobre el que se utilizará la aplicación soportará una versión de Android igual o superior a la 7.1.

RNF02. Facilidad de uso. El sistema deberá ser fácil de usar para cualquier usuario que utilice la aplicación.

RNF03. Navegación por la aplicación. La navegación por la interfaz debe ser muy sencilla, reduciendo el número de pantallas lo máximo posible.

RNF4. Control de múltiples dispositivos. El sistema tendrá la capacidad de soportar la integración de múltiples dispositivos inteligentes.

RNF05. Velocidad de respuesta. El sistema debe tener un tiempo de respuesta bajo entre la aplicación y el hardware integrado.

RNF06. Soporte multi-idioma. La aplicación debe tener la capacidad de mostrar los textos en español y en inglés.

RNF07. Persistencia de la información. El sistema debe tener la capacidad de almacenar las sesiones guardadas.

RNF08. Notificación de errores. El sistema debe proporcionar mensajes de error que sean informativos y útiles para el usuario.

RNF09. Sistema modular. El sistema se diseñará para que mejorar sus funcionalidades o añadir nuevas capacidades sea un proceso lo más sencillo posible.

3.3. Diseño de la aplicación

En esta sección se recogen las decisiones tomadas durante el diseño de la aplicación, se discuten las partes más destacables que componen la lógica del sistema y se refleja el diseño de la interfaz de usuario.

3.3.1. *Arquitectura del Sistema*

La aplicación desarrollada, *MyPlayRoom*, se ejecutará sobre un dispositivo Android (versión 7.1 o superior) con conectividad a Internet. Desde esta aplicación, se realizarán peticiones HTTP a un servidor de *Home Assistant* desplegado en una Raspberry Pi 4 Model B [21], tal y como se puede ver en la Figura 3.1.

Se realizará la integración de los dispositivos IoT en la plataforma de *HomeAssistant*, de tal forma que se establezca un canal de comunicación entre la plataforma y los dispositivos. Para realizar el descubrimiento de los dispositivos activos dentro de la red, *Home Assistant* utiliza el protocolo SSDP [22].

Una vez que este entorno está desplegado, desde *MyPlayRoom* se realizarán peticiones a la API REST de *Home Assistant* para desencadenar acciones en los dispositivos integrados.

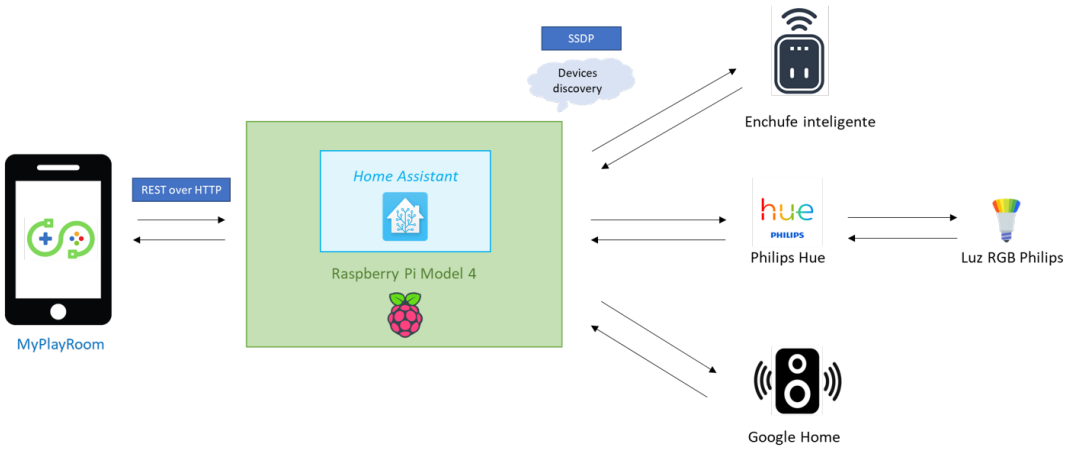

Figura 3.1: Arquitectura del Sistema

3.3.2. *Lógica del Sistema*

Con el objetivo de organizar a nivel lógico la funcionalidad definida y facilitar su desarrollo posterior, se ha desarrollado el diagrama de clases disponible en el **Anexo A**. En él se refleja la distribución de las principales clases diseñadas. Se ha decidido no incluir todas las clases implicadas en el proyecto para facilitar su comprensión, por lo que no se reflejan las clases dedicadas a la gestión de fragmentos,

elementos de la interfaz gráfica, ni excepciones.

A continuación, se describen todos los paquetes que conforman la aplicación, con una descripción de la funcionalidad de las clases que alberga cada uno de ellos:

- **Conexion.myPlayRoom**: contiene las clases dedicadas a la gestión de los fragmentos que componen *MyPlayRoom*, así como la clase *AppViewModel*, utilizada con el fin de almacenar y administrar datos relacionados con la interfaz de usuario de manera optimizada para la gestión de los ciclos de vida.
- **Homeassistant**: en este paquete se localizan las clases utilizadas para permitir la comunicación con *Home Assistant* y gestionar las peticiones realizadas al servidor.
- **Session**: alberga las clases dedicadas a la gestión de las sesiones creadas por los usuarios, conformadas por secuencias de acciones añadidas, distribuidas en los canales disponibles.
- **UI**: contiene las clases utilizadas para el desarrollo de la interfaz de usuario.
- **Exceptions**: en este paquete se han incluido las excepciones creadas para el control de errores de la aplicación: *ChannelAlreadyInUseException* y *DeviceAlreadyInUseException*.

3.3.3. *Diseño de la interfaz de usuario*

En esta subsección se muestran los prototipos de las pantallas diseñadas para la aplicación, junto a una descripción de cada una de ellas y de la navegación entre las mismas. Los diferentes fragmentos que compondrán la aplicación han sido diseñados para el cumplimiento de todos los requisitos funcionales y no funcionales recogidos en el apartado anterior.

La pantalla principal de la aplicación contendrá un botón a través del cual se realizará una llamada al servidor de *Home Assistant*, comprobando si está activo. En caso satisfactorio, el usuario podrá elegir dos modos de uso: *Play sesión* y *Grabar sesión*, como se muestra en la Figura 3.2. En caso contrario, se mostrará un mensaje de error y el usuario no podrá realizar ninguna otra acción hasta que se haya configurado correctamente el servidor.

Seleccionando la opción de *Play sesión*, se cargará un nuevo fragmento (ver Figura 3.3) en el que de nuevo se le dará al usuario la posibilidad de realizar dos acciones: pulsar sobre *Mis sesiones*, que permitirá la navegación al fragmento de la Figura 3.4, y seleccionar *PlayRoom*, que abrirá un nuevo fragmento a través del cual el usuario podrá interactuar con los dispositivos disponibles en tiempo real.

Desde el fragmento en el que se listan las sesiones disponibles, el usuario podrá: crear una sesión nueva, eliminar una sesión ya creada, o seleccionar una sesión en concreto para visualizar su contenido y poder modificarla, tal y como se puede observar en la Figura 3.4.

Si por el contrario el usuario utiliza el botón *Grabar sesión* de la pantalla inicial, representada en la Figura 3.5, se abrirá una ventana en la que el usuario deberá introducir el nombre de la sesión que

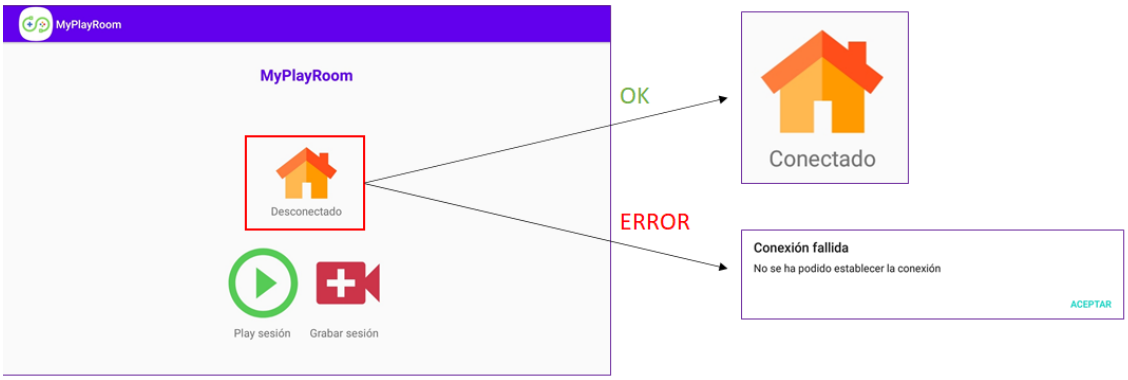

Figura 3.2: Pantalla principal - Comprobación de la conexión con *Home Assistant*

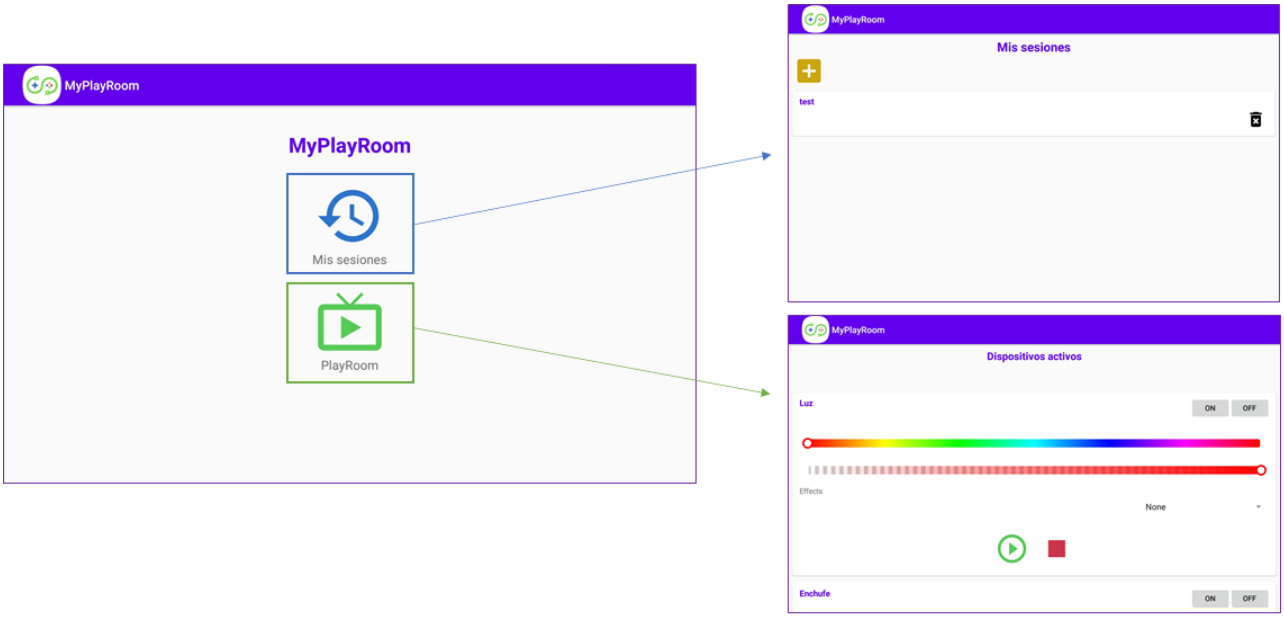

Figura 3.3: Pantalla *Play Sesión*

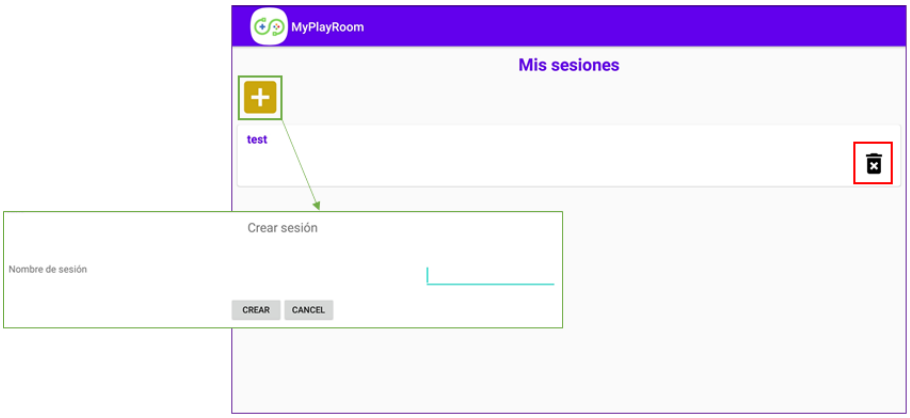

Figura 3.4: Pantalla listar sesiones

desea crear, y en caso de estar disponible dicho nombre (ya que se trata de identificativos únicos de sesión), se cargará el fragmento correspondiente a la sesión recientemente creada.

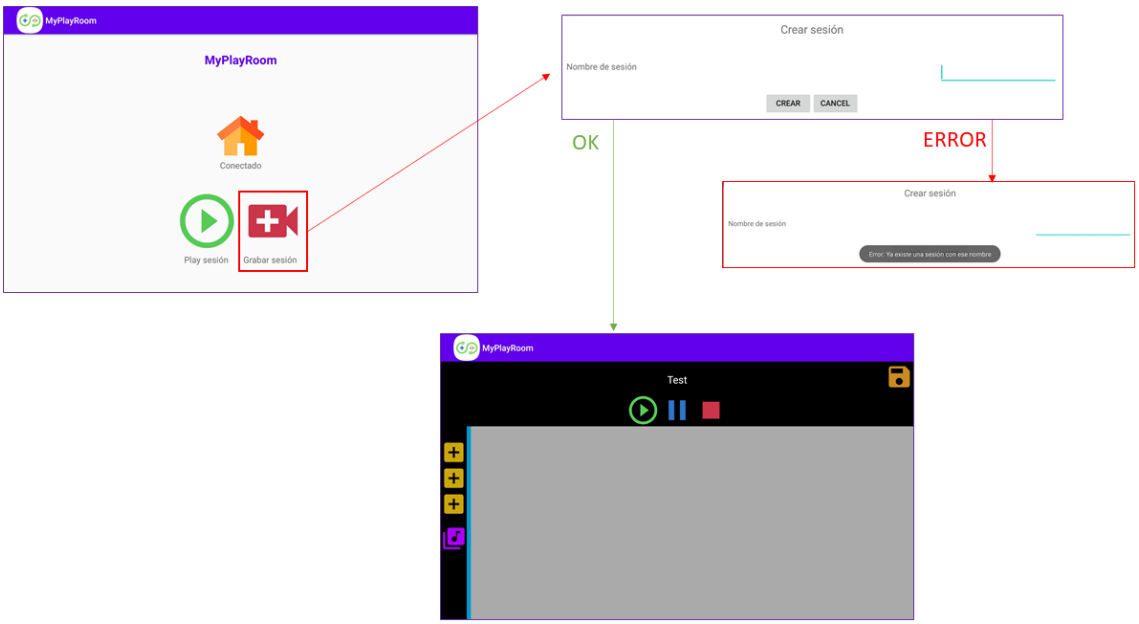

Figura 3.5: Pantalla grabar sesión

El fragmento de visualización de una sesión ilustrado en la Figura 3.6 estará compuesto de una barra en la parte superior con las acciones de *play*, *pause* y *stop*, un panel central en el que se podrá realizar *scroll* horizontal, de tal forma que se pueda visualizar el contenido de la sesión en su totalidad y el progreso de la ejecución a lo largo del tiempo; una barra vertical alojada en la parte izquierda de la pantalla desde la que se podrán añadir acciones a los canales disponibles (tres canales genéricos, y un canal de audio), y un botón de *Guardar* para almacenar los cambios realizados en el objeto sesión (ver Figura 3.7).

Cuando se seleccione uno de los botones disponibles para añadir una acción, se abrirá un cuadro como el de la Figura 3.8, en el que el usuario introducirá todos los elementos necesarios para añadir una acción: dispositivo, segundos en los que se desea que inicie y finalice la acción, la acción en cuestión, y los requisitos específicos de la acción en concreto (por ejemplo, canción que se desea reproducir, para la acción *play* de un dispositivo de tipo *MediaPlayer*).

Si la acción cumple todos los requisitos especificados para ser creada, se mostrará de nuevo el fragmento de visualización de sesión, con la nueva acción añadida representada. Esta representación consistirá en un bloque rectangular, de altura fija y ancho establecido en proporción a la duración de la acción, posicionada en el canal correspondiente y en el intervalo de tiempo seleccionado. En cuanto al contenido de este bloque, se mostrará el icono que representa al dispositivo en cuestión, y en el caso de la reproducción de sonidos (para dispositivos de tipo *MediaPlayer*), el icono que representa a la canción seleccionada. En el caso de los dispositivos de tipo *Light*, el fondo de dicho rectángulo será del color de la luz que se desea establecer y tendrá un fondo multicolor en caso de que se seleccione

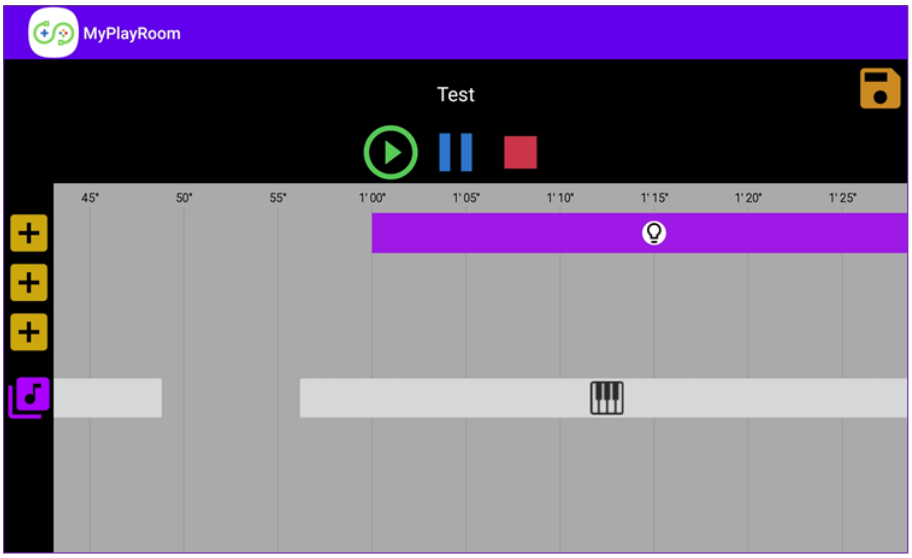

Figura 3.6: Pantalla de visualización de una sesión

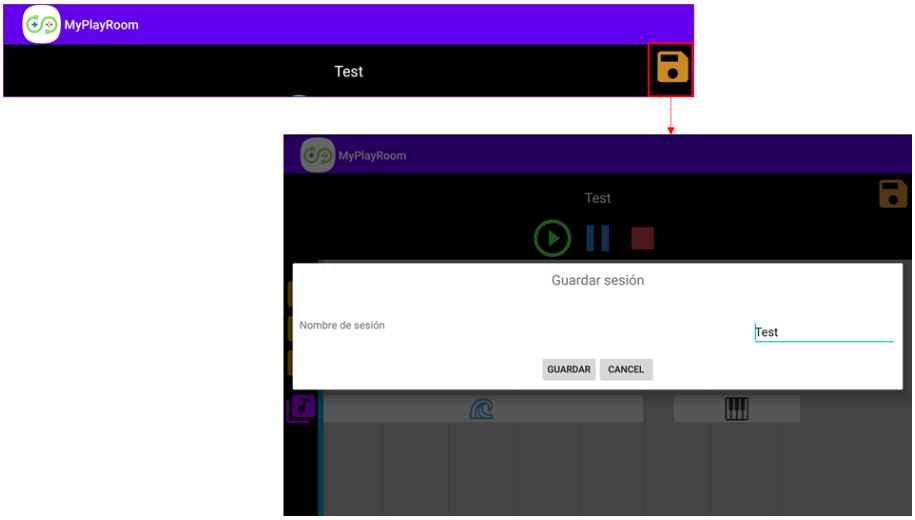

Figura 3.7: Pantalla guardar sesión

Figura 3.8: Pantalla añadir acción

el efecto *colorloop*. Para el resto de acciones, se utilizará un fondo gris genérico (ver Figura 3.9).

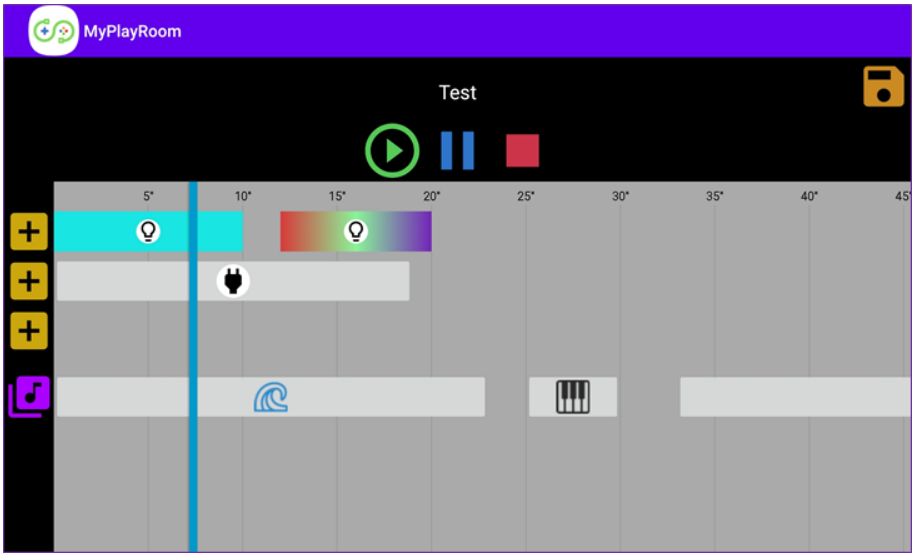

Figura 3.9: Pantalla sesión con acciones añadidas

El usuario además podrá eliminar acciones previamente creadas en una sesión haciendo click sobre los rectángulos que las representan. Esta interacción desencadenará la apertura de un nuevo fragmento como el de la Figura 3.10, en el que el usuario confirmará dicha eliminación y volverá al fragmento de la sesión.

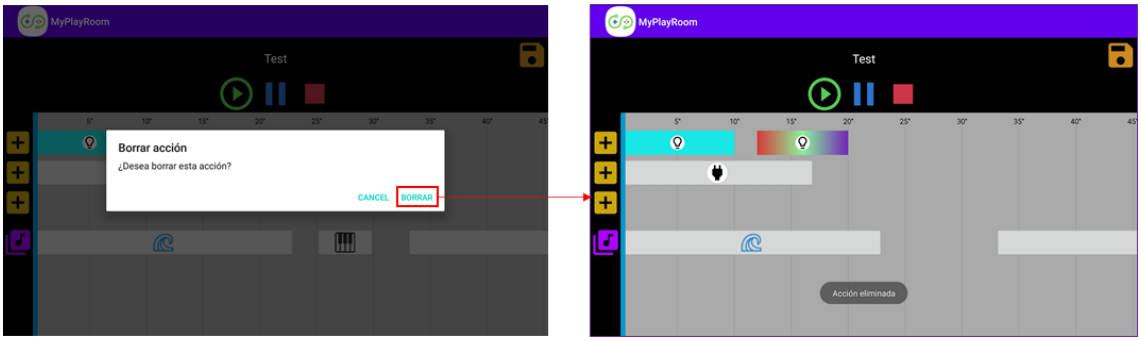

Figura 3.10: Pantalla eliminar acción

4

Despliegue

4.1. Despliegue de *Home Assistant*

Home Assistant proporciona cuatro métodos diferentes de instalación [23]. Para decidir qué opción se ajustaba más a las necesidades del proyecto, se ha realizado el siguiente análisis de cada una de ellas:

- **Home Assistant (antiguo *Hass.io*).** Incluye una distribución de Linux optimizada para ejecutar Home Assistant sobre *Docker* y un supervisor con varios contenedores Docker ya incluidos. Esta opción presenta las ventajas de ser muy sencilla y poseer un sistema de *Add-ons* muy potente. Como desventaja, no permite tener acceso real al sistema operativo del host que ejecuta el contenedor de Home Assistant (*Hass.io*).
- **Home Assistant Core.** Se trata del propio software como tal, que puede ser instalado mediante Entornos Python en cualquier distribución Linux e incluso en Windows. Existe la posibilidad de utilizar un contenedor *Docker* para alojar *Home Assistant Core*, lo que permitiría tener el control total sobre el sistema operativo *host*. No obstante, en ningún caso permite el empleo de *Add-ons* y *Snapshots*, lo cual supone una desventaja representativa.
- **Home Assistant Supervised.** Es una combinación de las dos opciones anteriores; permite utilizar un sistema operativo principal con *Docker* y *Home Assistant Core*, y además tener *Add-ons* y *Snapshots* con el *Supervisor*. Está actualmente mantenido por la comunidad *open source*, pero requiere un conocimiento avanzado para evitar problemas con su uso y asegurar una adecuada implementación.

Finalmente, se ha decidido instalar el sistema completo *Home Assistant* (*Hass.io)* en una Raspberry Pi, debido a su sencillez y al hecho de que posee un sistema de *Add-ons* muy potente; el hecho de no permitir tener acceso al sistema operativo del host que ejecuta el contenedor no supone un problema para la realización de este proyecto, dado que no se requieren configuraciones avanzadas en el *host*. En el **Anexo B** se pueden consultar los pasos seguidos para la instalación y adaptación de *Home Assistant* a los requisitos de este proyecto.

4.2. Integración de dispositivos

Una vez que el servidor de *Home Assistant* ha sido desplegado, se puede acceder a la interfaz web del servidor a través de la URL *http://homeassistant.local:8123*. Tras introducir las credenciales correspondientes se mostrará el panel de control, desde el que se podrán integrar los dispositivos utilizados en el prototipo de este proyecto. En el **Anexo C** se puede consultar más en detalle cómo se ha realizado la integración de cada uno de los dispositivos: la luz Philips Hue, el enchufe inteligente de Xiaomi y el dispositivo Google Home para ser utilizado como altavoz para la reproducción de sonidos.

Figura 4.1: Integración de dispositivos en *Home Assistant*

5

Implementación

5.1. Configuración del sistema

Para el correcto funcionamiento de la aplicación, se ha generado un fichero en formato *JSON* (ver Código 5.1), que alberga la siguiente información:

- **URL del servidor** de *Home Assistant* y **URL** en la que se localizan los **archivos de audio**.
- **Token** necesario para realizar la autenticación en las peticiones que se hacen desde la aplicación al servidor de *Home Assistant*.
- **Configuración de los dispositivos**; para cada uno de ellos se indica el tipo (*Light*, *Socket* o *MediaPlayer* en este prototipo), el *entity ID* del dispositivo, un *friendly name*, el nombre del icono que lo representa y dependiendo del dispositivo en concreto, más características que lo definen (por ejemplo, en el caso de las luces, la lista de efectos que se pueden utilizar).
- **Archivos de audio almacenados** en el servidor, incluyendo el nombre del archivo, el tipo (por si se desea asociar a un estado anímico o emoción para trabajar con el usuario), el *friendly name* que se mostrará en la aplicación y el icono que representará la acción en la que se utilice.

Esta configuración se carga en el método *onCreate* (Código 5.2) del fragmento inicial que se ejecuta al arrancar la aplicación, empleando la librería *gson* (ver Código 5.3). De esta forma, se crea un objeto *gson* e, invocando a su método *fromJson*, se le pasa como parámetros el objeto *JSON* como *String* y la clase del objeto en que se deserializará.

A partir de esta configuración leída, se crea de nuevo utilizando el mismo método un objeto de tipo *HAConnector*, que contiene el *token* y las URLs del servidor y de los archivos de audio (ver Código 5.4). A continuación, se instancian los dispositivos a partir de los valores del *haDeviceConfigs* y el *HAConnector* previamente obtenido (Código 5.5).

Además, en esta primera función ejecutada se deserializarán las sesiones almacenadas en el sistema para su recuperación.

Código 5.1: Fichero de configuración JSON

```
{
  "baseURL":"http://192.168.1.9:8123/api/",
  "mediaBaseURL": "http://192.168.1.9:8123/local/",
  "authToken": "eyJ0eXAiOiJKV1QiLCJhbGciOiJ[...]",
  "haDeviceConfigs": [
    {
      "type": "Light",
      "entityId": "light.luz",
      "friendlyName": "Luz",
      "icon": "icon_light_circle",
      "lightEffects": ["random", "colorloop"]
    },
    {
      "type": "Socket",
      "entityId": "switch.xiaomi_miio_switch",
      "friendlyName": "Enchufe",
      "icon": "icon_socket_circle"
    },
    {
      "type": "MediaPlayer",
      "entityId": "media_player.dormitorio",
      "friendlyName": "Altavoz",
      "icon": "icon_music_circle"
    }
  ],
  "haMediaConfigs": [
    {
      "name": "Ocean.mp3",
      "type": "relax",
      "friendlyName": "Ocean",
      "icon": "ic_wave"
    },
    [...]
}
```

Código 5.2: Método *onCreate* del fragmento inicial

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    appViewModel.config =
        HomeAssistantConfig.newInstance(requireContext().getString(R.string.configHA_file),
        requireContext())!!
    appViewModel.ha = HAConnector.fromConfig(appViewModel.config)
    createDevices()
    appViewModel.recoverSessions(requireContext())
}
```

Código 5.3: Instancia de *HomeAssistantConfig*

```
companion object {
    fun newInstance(file: String, context: Context): HomeAssistantConfig?{
        var gson = Gson()

        val jsonString: String
        return try {
            jsonString = context.assets.open(file).bufferedReader().use { it.readText() }
            gson.fromJson(jsonString, HomeAssistantConfig::class.java)
        } catch (ioException: IOException) {
            ioException.printStackTrace()
            null
        }
    }
}
```

Código 5.4: Instancia de *HAConnector* a partir de la configuración leída

```
companion object {
    fun fromConfig(config: HomeAssistantConfig): HAConnector{
        return HAConnector(config.baseURL, config.mediaBaseURL, config.authToken)
    }
}
```

5.2. Conexión con *Home Assistant*

De acuerdo a los requisitos definidos, se ha incluído una comprobación de que el servidor de *Home Assistant* está activo a la escucha de recibir peticiones antes de permitir al usuario navegar por la aplicación y realizar cualquier otra acción.

Al seleccionar este primer botón, se utiliza la función *testConnection* para comprobar dicho estado; si se recibe una respuesta por parte del servidor, se cambia el estado de la interfaz a *Connected* y se permite la navegación. En caso contrario, se abre una ventana emergente informando al usuario de que el servidor no está activo (ver Código 5.6).

5.3. Interacción con los dispositivos inteligentes

Una vez que los dispositivos han sido instanciados en la clase correspondiente de tipo *HADevice* (*Light, Socket o MediaPlayer*), tal y como se refleja en el apartado 5.1, cada dispositivo posee una lista de instancias de la clase *ActionableFunction* definidas al construir el dispositivo.

Cada instancia de *ActionableFunction* está compuesta por un *friendly name*, que se corresponde

Código 5.5: Proceso de creación de los dispositivos

```
fun createDevices(){
    appViewModel.haDevices =
        HADeviceFactory.instantiateFromConfig(appViewModel.config!!.haDeviceConfigs,
        appViewModel.ha)
}

//Mapeo config -> dispositivo
fun instantiateFromConfig(
    devicesConfig: List<HomeAssistantConfig.HADeviceConfig>,
    haConnector: HAConnector
    ): List<HADevice?> {
        return devicesConfig.map { HADeviceFactory.makeDevice(it, haConnector) }
    }

//makeDevice
 fun makeDevice(
        haDeviceConfig: HomeAssistantConfig.HADeviceConfig,
        haConnector: HAConnector
    ): HADevice? {

        return when (haDeviceConfig.type) {
            "Light" -> Light(
                haConnector,
                haDeviceConfig.entityId,
                haDeviceConfig.friendlyName,
                haDeviceConfig.icon,
                haDeviceConfig.lightEffects
            )
            "Socket" -> Socket(
                haConnector,
                haDeviceConfig.entityId,
                haDeviceConfig.friendlyName,
                haDeviceConfig.icon
            )
            "MediaPlayer" -> MediaPlayer(
                haConnector,
                haDeviceConfig.entityId,
                haDeviceConfig.friendlyName,
                haDeviceConfig.icon
            )

            else -> null
        }
    }
```

Código 5.6: Comprobación del estado del servidor de *Home Assistant*

```
override fun onResume() {
    super.onResume()

    //Botón comprobar estado conexión
    home_disconnected_btn.setOnClickListener{
        appViewModel.ha.testConnection{ result: Boolean ->
            if(result){
                onSuccesfulConectHASelected()
            }else{
                //lanzar aviso al usuario
                [...]
            }
        }
    }
    [...]
}

fun testConnection(ui_callback: (Boolean) -> Unit){
    //comprobar que la API está corriendo
    try{
        request("", "GET", JSONObject(), object: Callback {
            override fun onFailure(call: Call, e: IOException) {
                ui_callback(false)
                e.printStackTrace()
            }

            override fun onResponse(call: Call, response: Response) {
                response.use {
                    if (!response.isSuccessful)
                        ui_callback(false)

                    ui_callback(true)
                    println(response.body!!.string())
                }
            }
        })
    }catch(e: Exception){
        ui_callback(false)
    }
}
```

con el nombre de la función que verá el usuario en la interfaz, el *codeName* que refleja el nombre de la función en código, *opositeAction* para definir la clase semánticamente opuesta a la acción que representa, y la variable *params*, para describir los parámetros que son necesarios para realizar la petición *acción* al servidor y utilizados para crear distintos *widgets* en la interfaz (barra de color RGB, *spinners* que se muestran al usuario, etc.).

Código 5.7: Creación de un *ActionableFunction* para la función *setEffect* de la clase *Light*

```
1  //(Clase Light) ActionableFunction SetEffect
2  init {
3      [...]
4      actionableFunctions.add(
5          ActionableFunction(
6              "Effect",
7              "setEffect",
8              "turnOff",
9              params = mapOf(
10                 Pair("Color", ParameterType.COLOR),
11                 Pair("Effect", ParameterType.EFFECT)
12             )
13         )
14     )
15     [...]
16 }
```

Con el objetivo de ejemplificar en qué consiste el esquema de una llamada, se ha realizado el diagrama de secuencia de la Figura 5.1. La estructura de una petición (ver Código 5.8) consiste en lo siguiente:

1.– Desde la UI se utiliza el botón correspondiente, desde el que se llama a la función en cuestión (*rgbColor* en este ejemplo).

2.– La función *rgbColor* convierte sus parámetros de entrada a un objeto *JSON*, que se pasa a la función *request* de *HAConnector*, junto con la URL y el método HTTP.

3.– En la función *request* se realiza una petición HTTP al servidor de *Home Assistant*.

4.– La aplicación recibe la respuesta del servidor y ejecuta el *callback* que se mandó en la petición para permitir la gestión de errores.

5.4. Almacenamiento de sesiones

Para añadir las acciones que crea el usuario utilizando la interfaz a una sesión, se utiliza el método *addUserAction* (ver Código 5.9), el cual añade una acción a un canal determinado comprobando previamente que este canal está disponible en el rango de tiempo seleccionado y a su vez que el dispositivo no está siendo utilizado en otro canal.

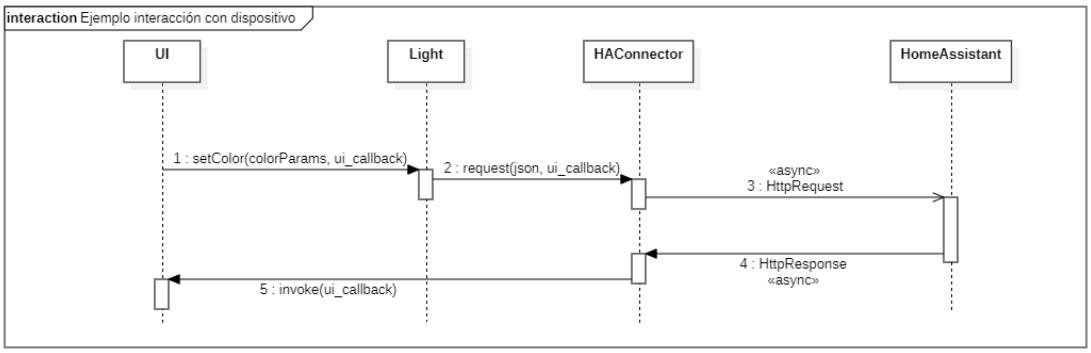

Figura 5.1: Diagrama de secuencia: Cambiar de color luz RGB

Código 5.8: Ejemplo llamada *rgbColor*

```
1  //Llamada desde la UI
2  override fun onColorChange(){
3      [...]
4      luz?.rgbColor(currentColor, {})
5  }
6
7  //Clase Light
8  fun rgbColor(colorParams: ColorParams, ui_callback: (Boolean) -> Unit) {
9
10     //creación del objeto JSON a partir de los valores de colorParams
11     [...]
12      //ejemplo request
13      haConnector.request("
```

Código 5.9: Función para añadir acciones a una sesión

```
1  //Clase Session
2  fun addUserAction(userAction: UserAction, channelId: Int) {
3      checkFreeChannel(channelId, userAction)
4      checkDeviceInUse(userAction)
5      channels[channelId]?.add(userAction)
6  }
```

Para guardar las sesiones creadas y que puedan ser recuperadas y reproducidas en un momento posterior a su creación, la clase *Session* y todas las clases de los elementos que la componen implementan la interfaz *Serializable*. De esta forma, se crea una representación binaria de todas las clases para que puedan ser almacenadas en un fichero y recuperadas en un futuro.

Las funciones creadas para dicho fin, *saveSessions* y *recoverSessions*, pueden ser consultadas en el Código 5.10.

Código 5.10: Almacenamiento y recuperación de sesiones

```
1  //Guardar sesiones
2  fun saveSessions(context: Context) {
3          context.openFileOutput(
4              context.getString(R.string.sessions_file),
5              Context.MODE_PRIVATE
6          ).use { output ->
7              var objectOutputStream = ObjectOutputStream(output)
8              objectOutputStream.writeObject(sessionsMap)
9          }
10     }
11 //Recuperar sesiones
12 fun recoverSessions(context: Context) {
13
14     try{
15         context.openFileInput(context.getString(R.string.sessions_file))
16             .use { input ->
17              var objectInputStream = ObjectInputStream(input)
18              sessionsMap = objectInputStream.readObject() as TreeMap<String, Session>
19             }
20
21         //caso: si cambia la IP de HomeAssistant tras recuperar sesion, se actualice
22         sessionsMap.forEach { it ->
23             it.value.channels.values.forEach {
24                 it.forEach { x ->
25                     x.actIni.haDevice.haConnector = ha
26                     x.actFin.haDevice.haConnector = ha
27                 }
28             }
29         }
30     }catch(e: FileNotFoundException){
31     sessionsMap = TreeMap()
32     }
33 }
```

5.5. Reproducción de secuencias de acciones

Cuando el usuario **inicia la reproducción** de una sesión, a la instancia de *SessionPlayer* se le comunica cuál es la sesión actual y se crea un hilo (implementado en la clase *SessionPlayerThread*),

encargado de mandar las peticiones configuradas en las acciones que conforman una sesión.

Cabe destacar que toda acción añadida por el usuario tiene asociada implícitamente tres acciones diferentes que serán ejecutadas según si la acción principal es pausada, reanudada o finalizada definitivamente. Por ejemplo, si la acción demandada consiste en reproducir un sonido, se le asocia de forma transparente para el usuario la acción *parar reproducción*, que se ejecutará cuando se pulse el botón *stop*. Del mismo modo ocurrirá cuando se pulse *pause*; se utilizaría en este caso la función *pausar reproducción* y al volver a pulsar el botón *play*, se ejecutaría *reanudar reproducción*.

La función que arranca la reproducción (ver el pseudocódigo 5.11) consiste en un bucle que se ejecuta cada 10ms (siendo percibido por el usuario como una ejecución en tiempo real). En dicho bucle, hasta que el usuario no finalice la reproducción (pulsando los botones *stop* o *pause*, acciones explicadas en detalle a continuación), se ejecuta cada acción pendiente y su acción opuesta (p.ej. si la acción consiste en encender una luz, al finalizar el periodo indicado se apagará), siempre que sean anteriores al *offset* actual y así hasta que finalice las acciones pendientes de reproducir.

Código 5.11: Pseudocódigo función que inicia la reproducción de una sesión: *run()*

```
1  //run()
2  inicializacion:
3      mPauseLock = Lock para que se quede el hilo bloqueado cuando pausamos
4      allSessionUserActions = [Todas las userActions de todos los canales de la sesion]
5      userActionsEnEjecución = []
6      tIni = timestampActual()
7
8  cada 10 ms:
9      if no flagStop:
10         t = timestampActual() -tIni # time offset actual
11
12         for each userAction in allSessionUserActions whose iniTimeOffset < t:
13             ejecutar userAction.actIni
14             añadir userAction a userActionsEnEjecución
15             quitar userAction de allSessionUserActions
16
17         for each userAction in userActionsEnEjecución whose finTimeOffset < t:
18             ejecutar userAction.actFin
19             quitar userAction de userActionsEnEjecución
20
21         if vacio(allSessionUserActions) y vacio(userActionsEnEjecución):
22             finalizado por peticion del usuario = True
23             continue
24
25         if flagPause: #flag cambiado en pause()
26             mPauseLock.wait() # Espera a que alguien haga mPauseLock.notify() --> (resumeThreat)
27             reajustar tIni para que se calcule correctamente t en la siguiente iteracion
```

La acción de **pause** (ver pseudocódigo 5.12) pausa el hilo (cambiando el *flagPause*) y para todas las acciones que se encontraban en ejecución, ejecuta la petición de *pausa*.

Código 5.12: Pseudocódigo función que pausa la reproducción de una sesión: *pause()*

```
1  //pause()
2  if (isStarted && (currentPlayerThread != null && currentPlayerThread!!.isAlive):
3      currentPlayerThread.pauseThread()
4      isPaused = true
5
6  //pauseThread()
7  for each action in userActionsEnEjecución:
8      accion.dispositivo.peticionOnPause.ejecutar() # necesario para pausar reproducción de sonidos
9      flagPause = true
```

Al **finalizar la reproducción** (ver pseudocódigo 5.13), primero se ejecutan las acciones de *stop* asociadas a cada una de las acciones en curso (p.ej. apagar una luz que estaba en ese momento activa) y se cambia el *flagStop* para permitir que el hilo finalice.

Código 5.13: Pseudocódigo función que finaliza la reproducción de una sesión: *stop()*

```
1  //stop()
2  if(isStarted && (currentPlayerThread != null && currentPlayerThread!!.isAlive)):
3      currentPlayerThread.stopThread()
4      isStarted = false
5
6  //stopThread()
7  for each action in userActionsEnEjecución:
8      accion.dispositivo.peticionOnStop.ejecutar() # necesario para finalizar todas las acciones
9      flagStop = true
10     resumeThread() # para desbloquear el hilo si estaba en pause
```

6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este Trabajo de Fin de Grado se propuso el diseño y desarrollo de un sistema de control para la creación de una sala de estimulación sensorial utilizando elementos de bajo coste, permitiendo la posibilidad de configurar sesiones compuestas por secuencias de acciones personalizadas por un terapeuta adaptándolas a las necesidades de su usuario, así como permitir la interacción directa con dichos dispositivos para que pueda ser directamente manipulada por el usuario final.

Para ello, primeramente se ha analizado la situación actual del mercado en lo relativo al desarrollo e implantación de salas de estimulación multisensorial. Actualmente, existen empresas que proporcionan soluciones con tecnologías muy avanzadas, completamente adaptables a las necesidades de cada usuario y fácilmente escalables, cuyo núcleo se compone de un software de control en el que se puede integrar una gran variedad de dispositivos. Sin embargo, se trata de soluciones con precios muy elevados, lo que dificulta su adquisición a pesar de la gran cantidad de beneficios a nivel terapéutico que aportan.

De esta observación surgió la idea de recrear un entorno inteligente fácilmente controlable; tras analizar diferentes herramientas, se decidió aprovechar las capacidades de la plataforma de código abierto *Home Assistant* para la integración de dispositivos de diferentes fabricantes y utilizar la REST API que proporciona para la interacción con dichos dispositivos, realizando las llamadas necesarias al servidor desde el software creado en este proyecto.

Una vez diseñada la arquitectura general del entorno, se analizaron los requisitos, tanto funcionales como no funcionales, que se deseaba que tuviera la aplicación creada en este proyecto. Del mismo modo, se diseñaron los prototipos de las pantallas que componen la aplicación, cubriendo todas las necesidades descritas en dichos requisitos. Finalmente se desarrolló la aplicación *MyPlayRoom*, un software de control compatible con dispositivos Android que proporciona una interfaz amigable que permite la interacción directa con diferentes dispositivos inteligentes y la programación y reproducción de sesiones.

6.2. Trabajo futuro

Como trabajo futuro, sería interesante desplegar este proyecto en un entorno real y realizar pruebas con usuarios finales para detectar posibles errores en la implementación, recoger sus impresiones y así poder mejorar las capacidades de la herramienta.

Además, tal y como está estructurado el código, sería relativamente fácil para un desarrollador incluir soporte para nuevos dispositivos, por lo que este prototipo podría ser utilizado como base y adaptado a otra serie de proyectos.

Bibliografía

[1] "¿Cuánto cuesta una sala multisensorial?." https://blog.bjadaptaciones.com/cuanto-cuesta-una-sala-multisensorial/. (Último acceso 10/05/2021).

[2] "El porqué de las salas multisensoriales shx para personas con tea." http://www.autismo.org.es/actualidad/articulo/el-porque-de-las-salas-multisensoriales-shx-para-personas-con-tea. (Último acceso 10/05/2021).

[3] "Doit, fabricantes de equipos sensoriales y salas multisensoriales." https://www.doitmultisensorial.com/en/. (Último acceso 17/05/2021).

[4] "Control de sala multisensorial." https://www.doitmultisensorial.com/en/sala_multisensorial_doit_console/. (Último acceso 17/05/2021).

[5] "Doit link - para el control de la sala multisensorial." https://www.doitmultisensorial.com/doit-link-tech/. (Último acceso 17/05/2021).

[6] "Catálogo de productos sensoriales y salas multisensoriales doit." https://www.doitmultisensorial.com/en/2018/11/19/catalogo-de-material-sensorial-doit/, Oct 2019. (Último acceso 17/05/2021).

[7] "Espacios sensoriales." https://www.eneso.es/que-ofrecemos/espacios-sensoriales/. (Último acceso 17/05/2021).

[8] "Eneso sense, tecnología para salas de estimulación multisensorial." https://www.eneso.es/blog/eneso-sense-tecnologia-salas-estimulacion-multisensorial/. (Último acceso 17/05/2021).

[9] "Shop." https://www.eneso.es/shop/category/espacios-sensoriales. (Último acceso 17/05/2021).

[10] "Bj adaptaciones." https://bjadaptaciones.com/. (Último acceso 17/05/2021).

[11] "Shx rack." https://bjadaptaciones.com/42-sistema-shx. (Último acceso 17/05/2021).

[12] R. Pi, "Buy a raspberry pi." https://www.raspberrypi.org/products/. (Último acceso 17/05/2021).

[13] "Empowering the smart home." https://www.openhab.org/. (Último acceso 17/05/2021).

[14] . Soloelectronicos, "6 herramientas de domótica de código abierto." https://soloelectronicos.com/2019/11/14/6-herramientas-de-domotica-de-codigo-abierto/, Nov 2019. (Último acceso 17/05/2021).

[15] W. Beaton, "Eclipse public license 2.0: The eclipse foundation." https://www.eclipse.org/legal/epl-2.0/. (Último acceso 17/05/2021).

[16] H. Assistant, "Home assistant." https://www.home-assistant.io/, May 2021. (Último acceso 17/05/2021).

[17] "Rest api: Home assistant developer docs." https://developers.home-assistant.io/docs/api/rest/. (Último acceso 17/05/2021).

[18] "Iluminación inteligente." https://www.philips-hue.com/es-es. (Último acceso 17/05/2021).

[19] "Google home altavoz." https://www.mediamarkt.es/es/product/_altavoz-inteligente-asistente-google-home-smart-home-dom%C3%B3tica-bluetooth-sonido-360%C2%BA-tiza-1421130.html. (Último acceso 17/05/2021).

[20] "Xiaomi mi smart wifi socket plug zncz02cm." https://xiaomi-pedia.com/unboxing-xiaomi-mi-smart-wifi-socket-plug-zncz02cm.html, Oct 2016. (Último acceso 17/05/2021).

[21] R. Pi, "Raspberry pi 4 model b." https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. (Último acceso 17/05/2021).

[22] "SSDP protocol." https://wiki.wireshark.org/SSDP. (Último acceso 17/05/2021).

[23] D. M. González, "Domotizando nuestra casa con home assistant." https://www.nocountryforgeeks.com/domotizando-nuestra-casa-con-home-assistant/, Jul 2019. (Último acceso 17/05/2021).

Apéndices

A

Diagrama de clases de la aplicación

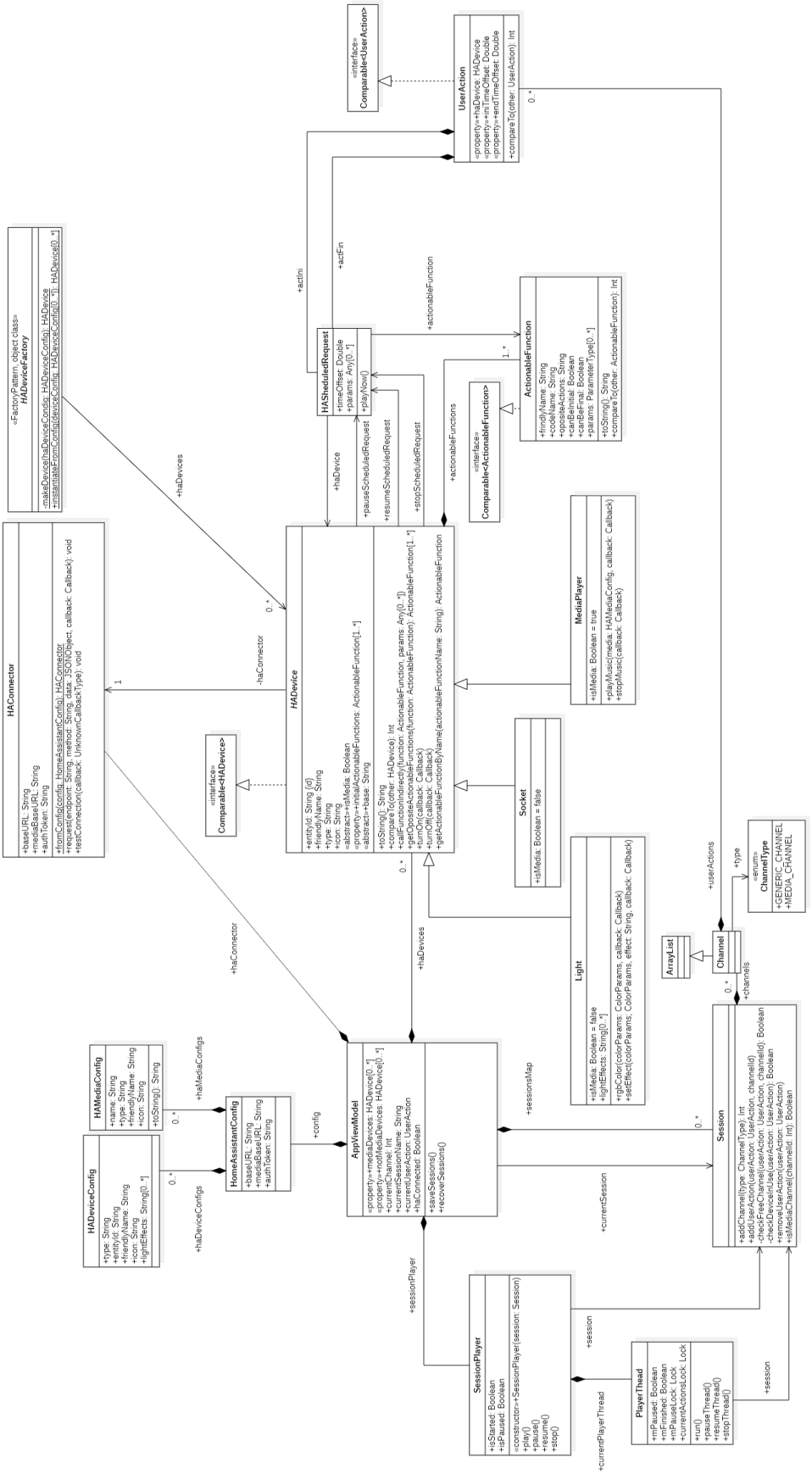

Figura A.1: Diagrama de las clases más representativas que conforman *MyPlayRoom*

B

Instalación y despliegue de *Home Assistant*

B.1. Requisitos para la instalación

Para poder realizar la instalación y despliegue del servidor de *Home Assistant*, se ha requerido la utilización de los siguientes elementos hardware:

- Raspberry Pi 4 Model B
- Fuente de alimentación de 3A.
- Tarjeta microSD 16GB.
- Lector de tarjetas SD.
- Adaptador de tarjetas MicroSD/SD.
- Cable Ethernet.

B.2. Descarga de la imagen

1.– Utilizando el adaptador MicroSD/SD, introducir la tarjeta MicroSD en el ordenador desde el que se va a cargar la imagen.

2.– Descargar y ejecutar el programa *Balena Etcher* (`https://www.balena.io/etcher/`).

3.– Seleccionar la opción *Flash from URL*.

4.– Pegar la siguiente dirección en el cuadro de texto *Use Image URL*: `https://github.com/home-assistant/operating-system/releases/download/5.13/hassos_rpi4-64-5.13.img.xz` de la Figura B.1. En el caso de que se utilice otro modelo de Raspberry, se deberá consultar en la documentación oficial cuál es la URL de descarga correspondiente.

5.– Cuando se complete la descarga de la imagen, hacer click sobre el botón *Select target*.

6.– Seleccionar la tarjeta MicroSD que se va a utilizar en la Raspberry Pi (ver Figura B.2).

7.– Hacer click en *Flash!* para que comience la escritura de la imagen.

8.– Cuando finalice el proceso, se mostrará una pestaña como la de la Figura B.3.

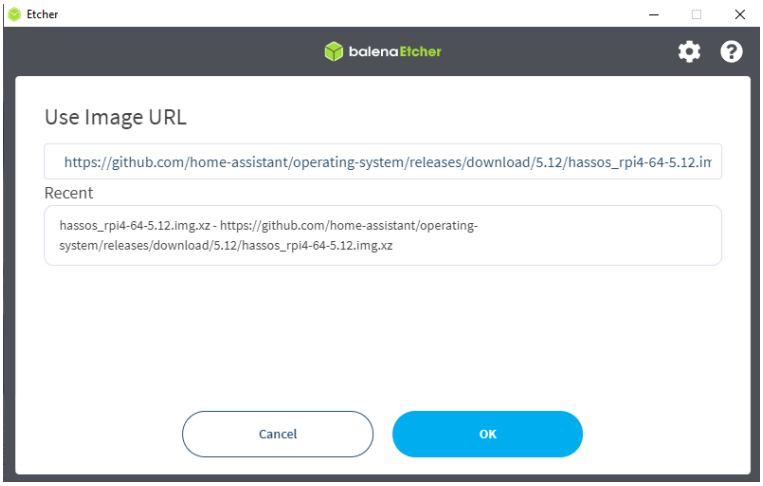

Figura B.1: Descarga de la imagen

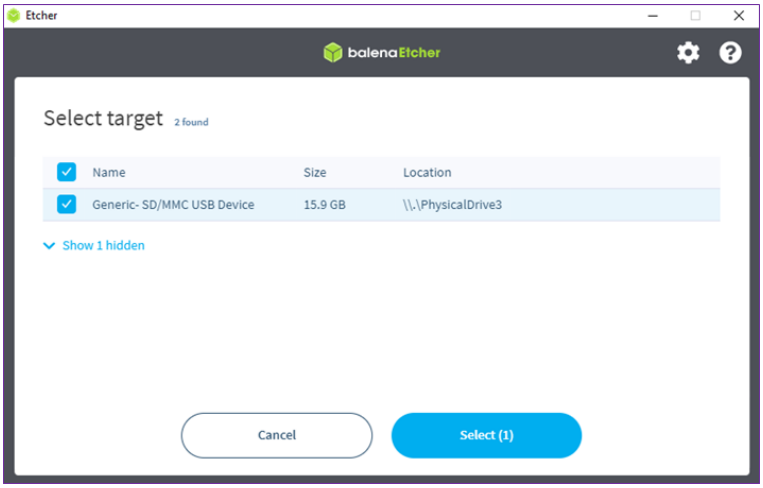

Figura B.2: Selección de la tarjeta SD

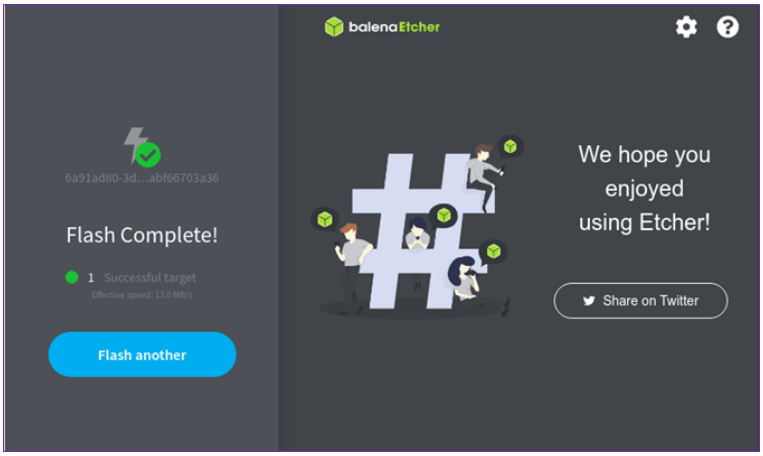

Figura B.3: Finalización del proceso

B.3. Puesta en marcha de la Raspberry Pi

A continuación, es necesario seguir estos pasos para configurar la imagen recientemente descargada:

1.– Conectar la fuente de alimentación de la Raspberry Pi.

2.– Insertar la tarjeta MicroSD.

3.– Conectar la Raspberry Pi a la red, utilizando para ello un cable Ethernet.

4.– Tras un par de minutos, *Home Assistant* será accesible desde la URL `http://homeassistant.local:8123`, o en su defecto en la URL `http://X.X.X.X:8123`, donde las X se corresponden con la dirección IP de la Raspberry.

B.4. Configuración de *Home Assistant*

Una vez instalado, el servidor será accesible a través de la siguiente URL: `http://homeassistant.local:8123`. Para continuar con la configuración:

1.– Crear una cuenta de usuario.

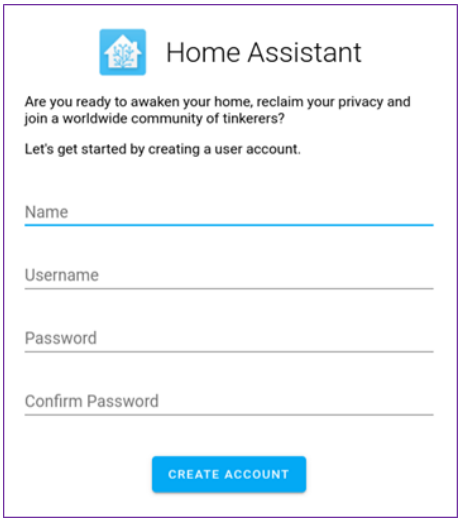

Figura B.4: Creación de una cuenta de usuario

2.– Introducir el nombre que se desea asociar a la cuenta.

3.– Hacer click en *Detect* para que en base a la localización actual se detecte automáticamente la localización y el sistema de medida (Figura B.5).

4.– Al hacer click en *Next*, se mostrarán los dispositivos descubiertos dentro de la red. Omitir este paso para integrar los dispositivos en un futuro y hacer click en *Finish* (Figura B.6).

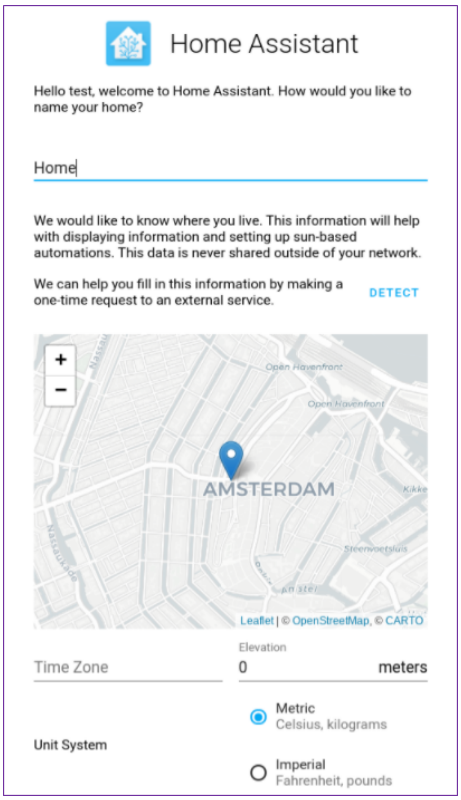

Figura B.5: Nombre de la cuenta

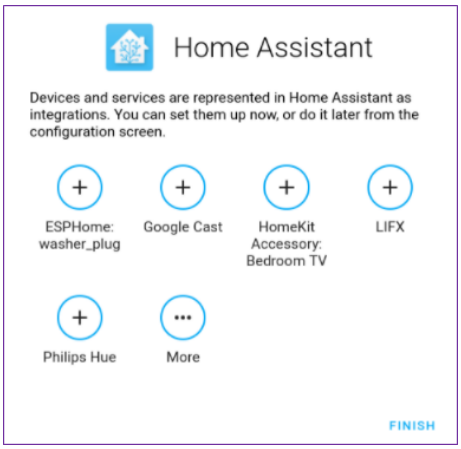

Figura B.6: Finalizar configuración de *Home Assistant*

C

Integración de dispositivos

A continuación se detallan los pasos seguidos para realizar la integración de los tres dispositivos incluidos en el prototipo de este proyecto. La integración se ha realizado aprovechando la interfaz web de *Home Assistant*, accesible desde la URL `http://homeassistant.local:8123` una vez que el servidor ha sido configurado y está activo.

C.1. Luz Philips Hue

1.– El primer paso consiste en conectar el Philips Hue a la red local utilizando un cable Ethernet.

2.– Al acceder a la URL `http://homeassistant.local:8123` se muestra la interfaz de la Figura C.1. En el apartado de ***Notifications*** se puede observar una nueva alerta. Al hacer click sobre la misma, se abre un cuadro de diálogo en el que se informa al usuario de que se han descubierto nuevos servicios dentro de la red local, disponibles para ser integrados. Hacer click sobre ***Check it out*** para continuar.

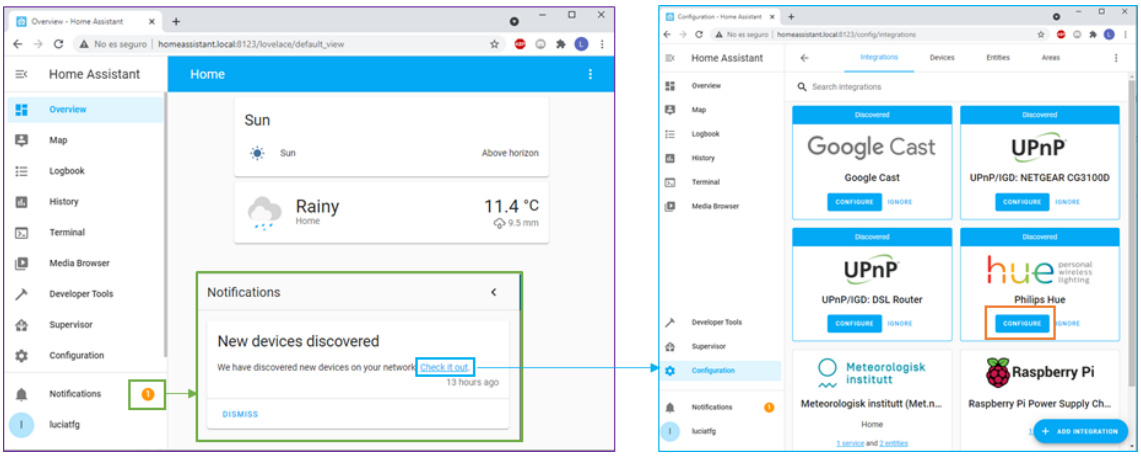

Figura C.1: Integración Luz Philips: Descubrimiento del dispositivo

3.– Se abre la pestaña de *Configuration* >*Integrations*, en la que se debe pulsar el botón ***Configure*** del cuadro correspondiente a la Luz Philips (ver Figura C.1).

4.– Se inicia la instalación de la integración y pasados unos segundos se pide al usuario **introducir la IP** del *Hue*. Es posible que *Home Assistant* lo detecte automáticamente. En caso contrario, será necesario consultar la IP asignada al dispositivo (ver Figura C.2).

5.– A continuación, es necesario **Pulsar el botón central** del dispositivo *Philips Hue* y pulsar sobre ***Submit*** (ver

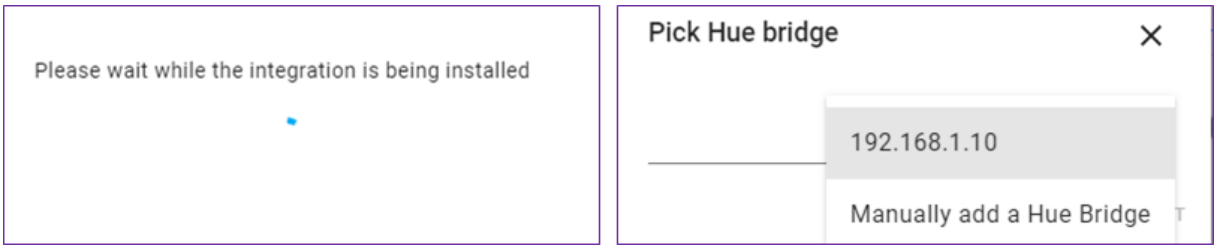

Figura C.2: Integración Luz Philips: Instalación en curso

Figura C.3). **NOTA:** Es importante saber que tras pulsar el botón, no se producirá ningún cambio hasta que se pulse el botón *Submit*.

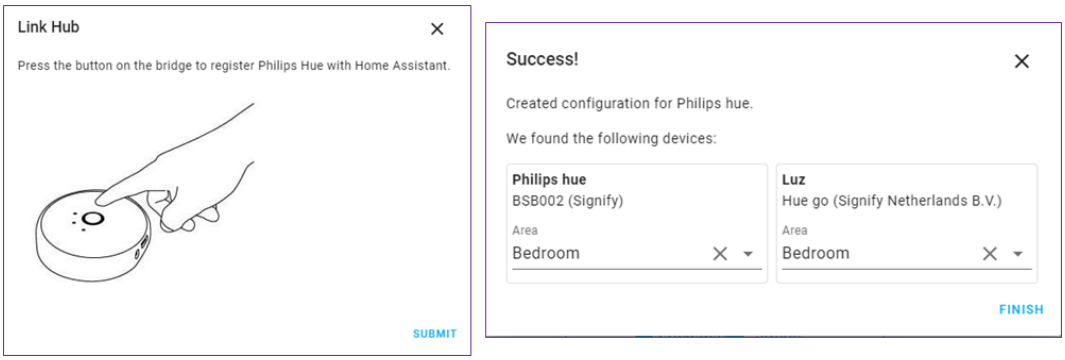

Figura C.3: Integración Luz Philips: Configuración del *Hue*

6.– Una vez finaliza el proceso, se muestra al usuario un cuadro de texto en el que se informa de que se han encontrado en la red dos dispositivos: el *Philips Hue* y la *Luz* conectada. En este caso se han localizado ambos dispositivos en el mismo área, *Bedroom*.

7.– Finalmente, se puede observar el nuevo dispositivo integrado en la pestaña de *Overview* (ver Figura C.4).

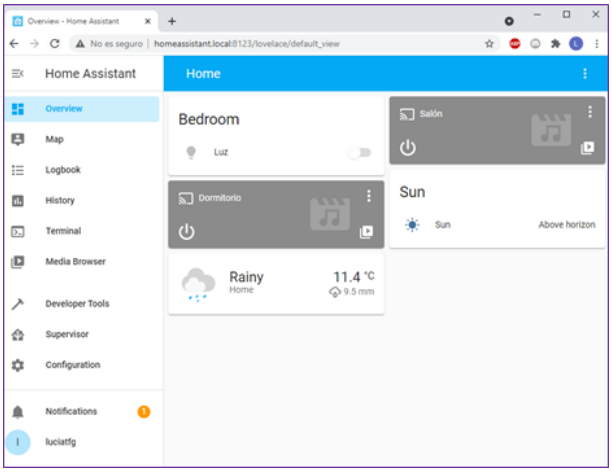

Figura C.4: Integración Luz Philips: Fin del proceso

C.2. Google Home

1.– Antes de comenzar con la integración, se debe conectar a la alimentación el dispositivo *Google Home* y comprobar que está conectado a la red local en la aplicación *Google Home* (ver Figura C.5).

2.– A continuación, en la pestaña *Configuration* >*Integrations*, hacer click sobre el botón ***Configure*** de *Google Cast*.

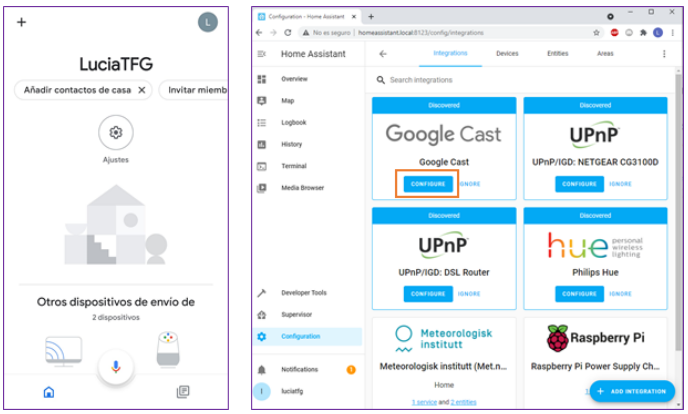

Figura C.5: Integración Google Home: Descubrimiento del dispositivo

3.– Se abre una ventana emergente, en la que se pide la confirmación del usuario para continuar con la integración. Hacer click en ***Submit***. A continuación, es necesario rellenar el área en el que se desea localizar el dispositivo *Google Home*. En este ejemplo, se ha escogido el área *Bedroom*.

NOTA: como se puede observar en la Figura C.6, *Home Assistant* también ha detectado la presencia de un dispositivo *Google Chromecast* en la red. Para este proyecto no es relevante su integración.

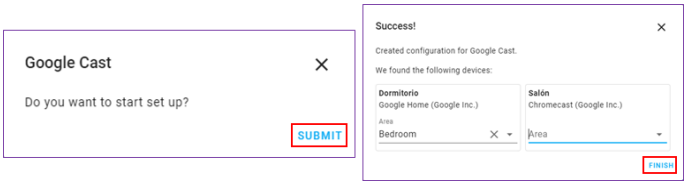

Figura C.6: Integración Google Home: Confirmación del usuario y asignación de un área

4.– Una vez finalizados estos pasos, se puede observar el nuevo dispositivo integrado en la pestaña de *Overview* (ver Figura C.7).

C.3. Enchufe inteligente Xiaomi

1.– El primer paso consiste en conectar en un enchufe el dispositivo, para lo que se ha utilizado un adaptador universal.

2.– Antes de comenzar con la integración en *Home Assistant*, es necesario descargar la aplicación de **Xiaomi *Mi Home*** (ver Figura C.8).

3.– Al iniciar la aplicación, se pide la **creación de una cuenta de usuario**. Una vez iniciada sesión, es importante **cambiar la región a China**, ya que el enchufe integrado es de origen chino. Esta acción se podrá realizar desde la pestaña de *Settings* >*Region* (ver Figura C.9).

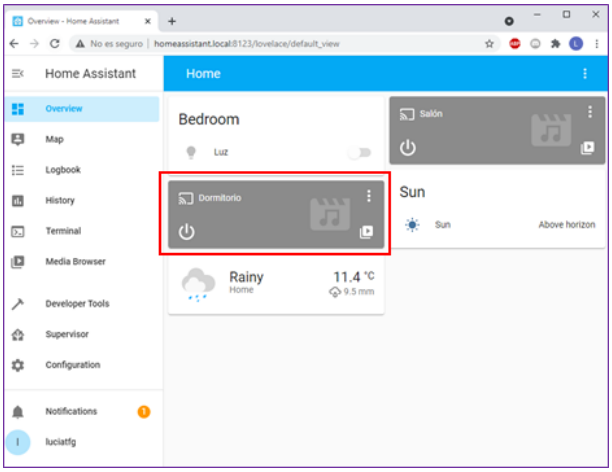

Figura C.7: Integración Google Home: Fin del proceso

Figura C.8: Integración Enchufe Xiaomi: Descarga de la aplicación *Mi Home*

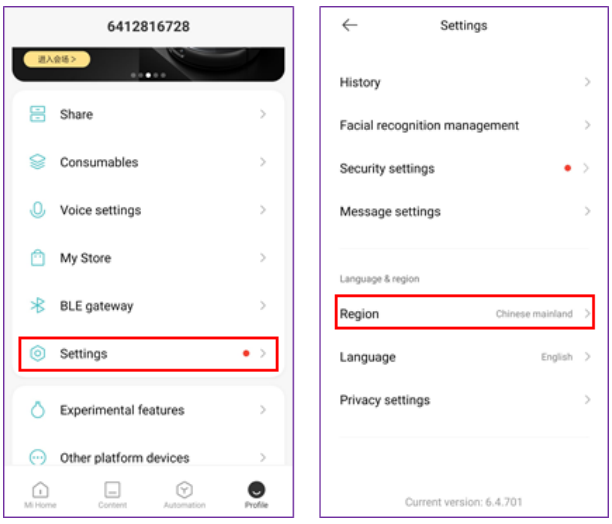

Figura C.9: Integración Enchufe Xiaomi: Cambio de región a China

4.– A continuación, desde la pestaña principal se inicia el escaneo de dispositivos. Para ello es necesario **aceptar los permisos** requeridos para la activación del Bluetooth y de la localización. Una vez finaliza el escaneo, es posible detectar el dispositivo *Mi Plug Mini* (ver Figura C.10).

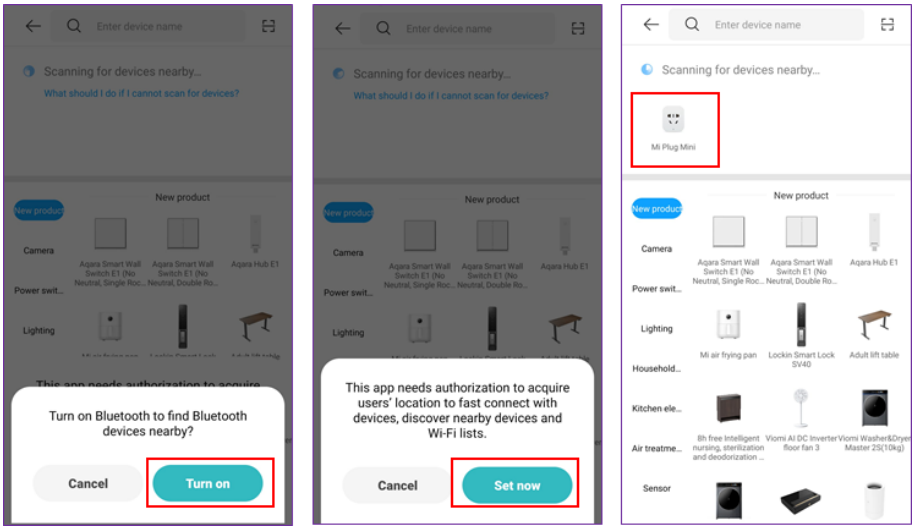

Figura C.10: Integración Enchufe Xiaomi: Escaneo de dispositivos

5.– Hacer click sobre el dispositivo y seguir los pasos requeridos para conectarlo al router (ver Figura C.11). Tras introducir la clave del Wi-Fi, pulsar sobre *Conectar* para permitir usar el dispositivo con *Xiaomi Home*.

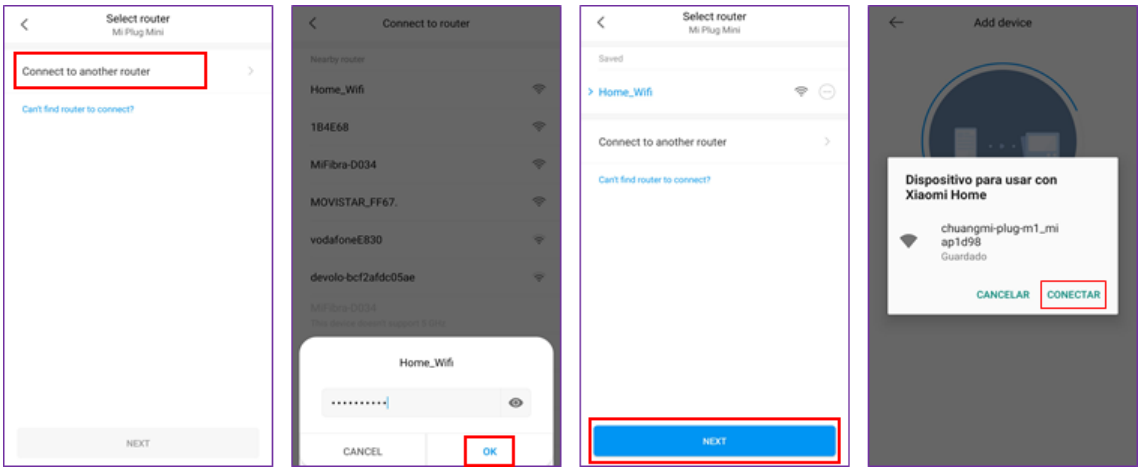

Figura C.11: Integración Enchufe Xiaomi: Conexión

6.– Volviendo a la interfaz de *Home Assistant* (`http://homeassistant.local:8123`), navegar a la pestaña *Configuration* >*Integrations* y pulsar sobre el botón ***Add Integration***.

7.– En el buscador, introducir *Xiaomi Miio*. Tras seleccionar la opción, se abrirá un nuevo cuadro de texto en el que se pide introducir la dirección IP y un API Token (ver Figura C.12).

8.– Para obtener el API token y la dirección IP del dispositivo:

8.1.– Descargar y descomprimir en la carpeta deseada el siguiente código: `https://github.com/PiotrMachowski/Xiaomi-cloud-tokens-extractor`.

8.2.– Ejecutar el archivo `token_extractor.exe`. Se deberán introducir las credenciales creadas en la aplicación *Mi Home* e introducir la región *cn* (China). A continuación, se mostrará el token necesario y la IP del dispositivo (ver Figura C.13).

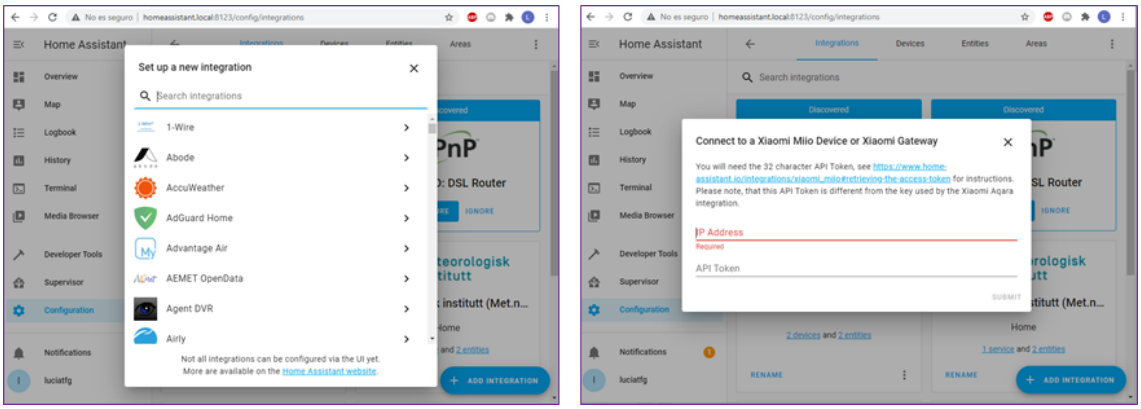

Figura C.12: Integración Enchufe Xiaomi: Añadir dispositivo

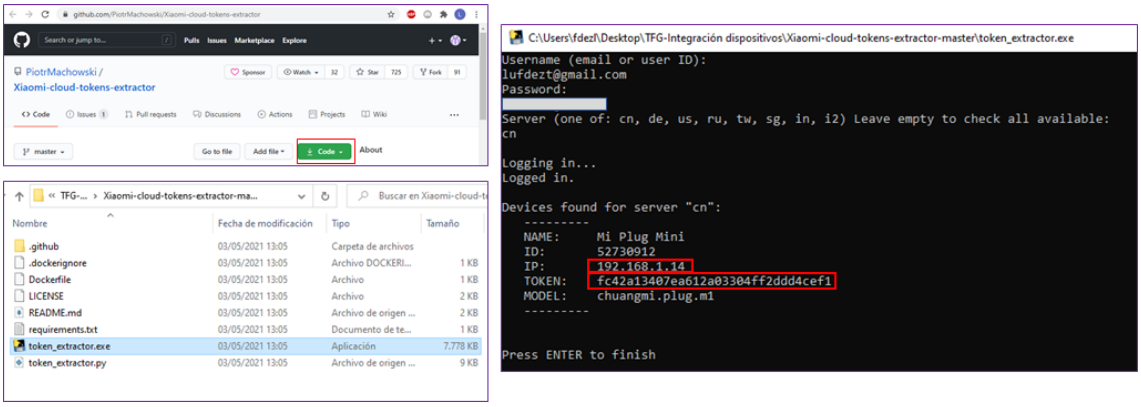

Figura C.13: Integración Enchufe Xiaomi: Extracción del token

9.– Una vez obtenidos ambos valores, se deben introducir en la ventana de *Home Assistant*. Es posible que se produzca un fallo durante el proceso de integración. En ese caso, se debe introducir el modelo del dispositivo *chuangmi.plug.m1* (ver Figura C.14).

10.– Una vez se complete este paso, se mostrará una ventana emergente informando al usuario. Pulsar sobre ***Submit*** para finalizar.

11.– En la pestaña *Overview* de *Home Assistant* ya pueden observarse todos los dispositivos integrados (ver Figura C.15.

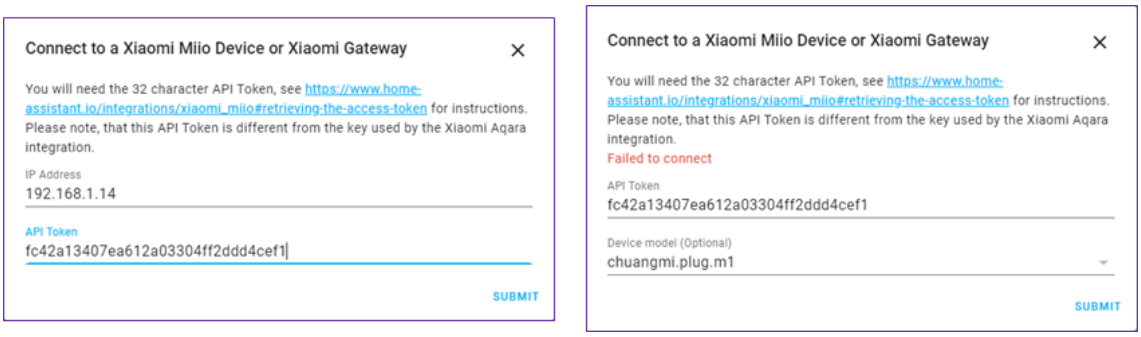

Figura C.14: Integración Enchufe Xiaomi: Introducción de los valores requeridos

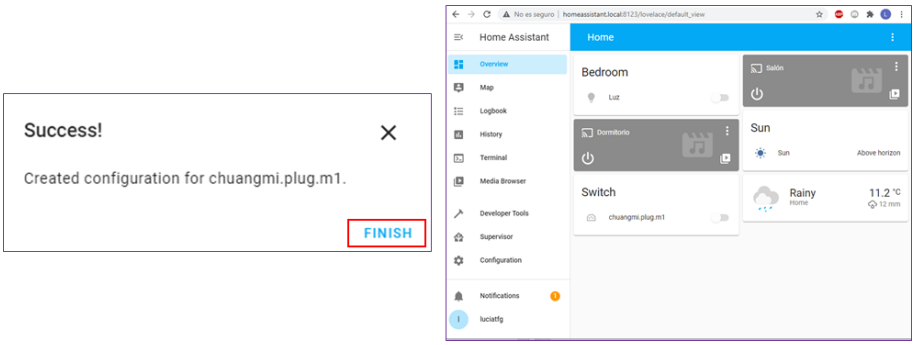

Figura C.15: Integración Enchufe Xiaomi: Fin del proceso


UAM
UNIVERSIDAD AUTONOMA
DE MADRID